Examination System Database Project Documentation

Prepared By

Ahmed Alaa Hassouna

David Neil Wilson

Rehab Ramadan Mohamed

Mahmoud Saad abdalkareem

Ali Abdallah Ali

Rawan Elsayed Mohamed

Supervisor :

Eng / Sarah

Examination System Database Project Documentation

1- Project Overview

Project Name	**Examination System Database**
Objective	• To design and implement a normalized, scalable, and secure database system that automates exam creation, management, and grading. • To provide role-based access for Managers, Instructors, and Students ensuring that each role can only access relevant tasks. • To maintain a question pool that supports random or manual exam generation. • To automatically evaluate objective questions and facilitate semi-automated/manual grading for text-based questions. • To ensure data integrity, consistency, and performance through indexing, constraints, triggers, and stored procedures. • To support daily backups and recovery mechanisms for data safety.
Scope	• Users: Training Managers, Instructors, Students (with separate roles and permissions). • Core Functions: Course management, exam creation, question pooling, exam scheduling, answer storage, grading, and result reporting. • Technical Coverage: Database normalization, indexing, integrity enforcement (constraints, triggers), and automated daily backup. • Limitations: The project focuses only on the database layer (schema, logic, data integrity, and security). • It does not include a full front-end application or external LMS integration.

2- Requirements Analysis

Functional Requirements

Question Management	• **Provide a question pool** for instructors to pick questions. • Support question types: Multiple Choice, True/False, and Text. • **For MCQ & TF questions** → Store correct answers, check student answers, and record results. • **For Text questions** → Store a best-accepted answer, use text functions & regex to validate, show valid/invalid answers to instructors for manual review and marking. • **Instructor can:** **1-** Create exams for their own courses only. Select questions manually or let the system pick randomly. **2-** Assign marks for each question, ensuring total does not exceed course's max degree. • **Exams must store details:** type (exam/corrective), intake, branch, track, course, start time, end time, total time, allowance options. • **System must store** exams by year, course, and instructor. • **Instructors can** assign students to specific exams and define exam time windows. • **Students can** see and attempt exams only during the defined time. • **System must store** student answers, check correctness, and compute final course result.
Course, Instructor, and Student Management	• **Course Information:** Store course name, description, maximum degree, and minimum degree. • **Instructor Information:** Store instructor details. An instructor can teach one or more courses. A course may be taught by one instructor per class (instructor may change for other classes/years). • **Training Manager Functions:** •Add and edit branches. •Add and edit tracks within each department. •Add new intakes. •Add students and define their personal data, intake, branch, and track.

Functional Requirements

User Accounts & Access Control	• **Each user (Training Manager, Instructor, Student)** must have a login account.
	• **Four account types:**
	  ◦ Admin → Perform admin tasks only.
	  ◦ Training Manager → Manage branches, tracks, intakes, students.
	  ◦ Instructor → Create/manage exams, define question degrees, review text answers.
	  ◦ Student → Take exams only at the scheduled time.
	• **Each account** can only access tasks related to their role

Exam Management	**Exam Creation (Instructor):** • Instructors can create exams for their courses only. • Select questions randomly or manually from the question pool. • Assign a degree for each question on the exam. • Total degrees for an exam must not exceed the course's maximum degree. • A course may have more than one exam. **Exam Details:** Store exam type (exam or corrective), intake, branch, track, course, start time, end time, and total time. **Exam Storage:** Store each defined exam by year, course, and instructor. **Student Exam Assignment:** Instructors can select students for specific exams and define the exam date, start time, and end time. **Student Exam Access:** Students can only see and take the exam during the specified time. **Answer Storage & Result Calculation:** Store student answers, calculate correct answers, and compute the final result for the student in the course.

Requirements Analysis

NonFunctional Requirements

Database Implementation	• **File Organization:** Implement the database using files and file groups based on data size and estimation. • **Data Types:** Choose appropriate data types for each column. • **Naming Conventions:** Use consistent naming conventions for all database objects. • **Indexing:** Implement indexes for optimal database performance. • **Data Integrity:** Use constraints and triggers to ensure data integrity and user access control. • **Stored Procedures & Functions:** Use procedures and functions for all system tasks. Use views to display results, eliminating the need for users to write direct queries.
Backup	• **Daily Backup:** The system must perform a daily backup automatically.

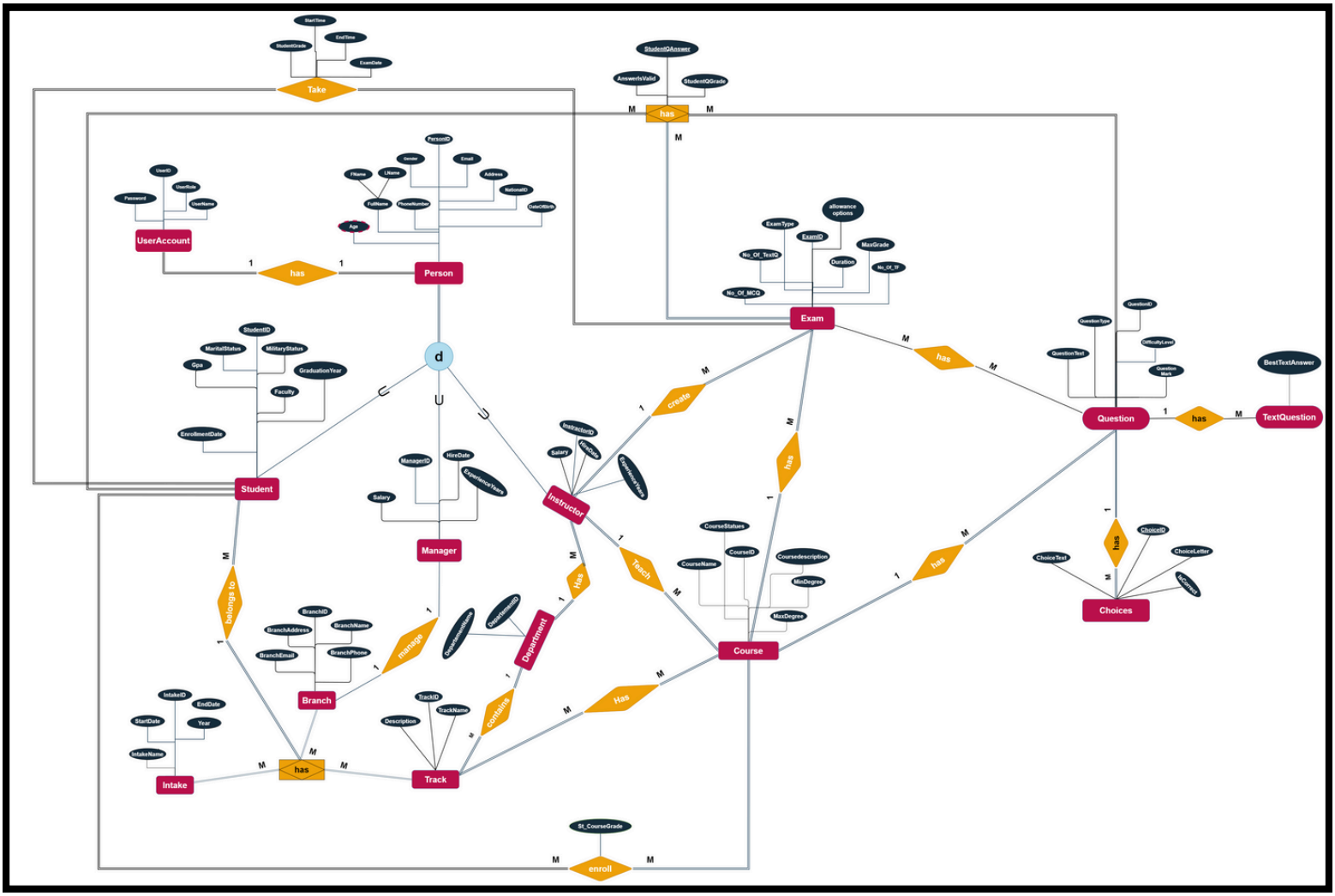
3.1 Main Entities

- **UserAccount** (UserID, UserRole, UserName, Password)
- **Person** (PersonID, FName, LName, PhoneNumber, Gender, Email, Address, NationalID, DateOfBirth)
- **Department** (DepartmentID, DepartmentName)
- **Branch** (BranchID, BranchName, BranchAddress, BranchEmail, BranchPhone)
- **Manager** (ManagerID, Salary, HireDate, ExperienceYears)
- **Student** (StudentID, MaritalStatus, GPA, MilitaryStatus, Faculty, EnrollmentDate, GraduationYear)
- **Instructor** (InstructorID, Salary, HireDate, ExperienceYears)
- **Intake** (IntakeID, IntakeName, StartDate, EndDate, Year)
- **Track** (TrackID, TrackName, Description)
- **Course** (CourseID, CourseName, CourseDescription, MinDegree, MaxDegree, CourseStatus)
- **Exam** (ExamID, ExamType, Duration, No_Of_MCQ, No_Of_TextQ, No_Of_TFQ, MaxGrade, AllowanceOptions)
- **Question** (QuestionID, QuestionType, QuestionText, DifficultyLevel, QuestionMark)

3.2 Relationships

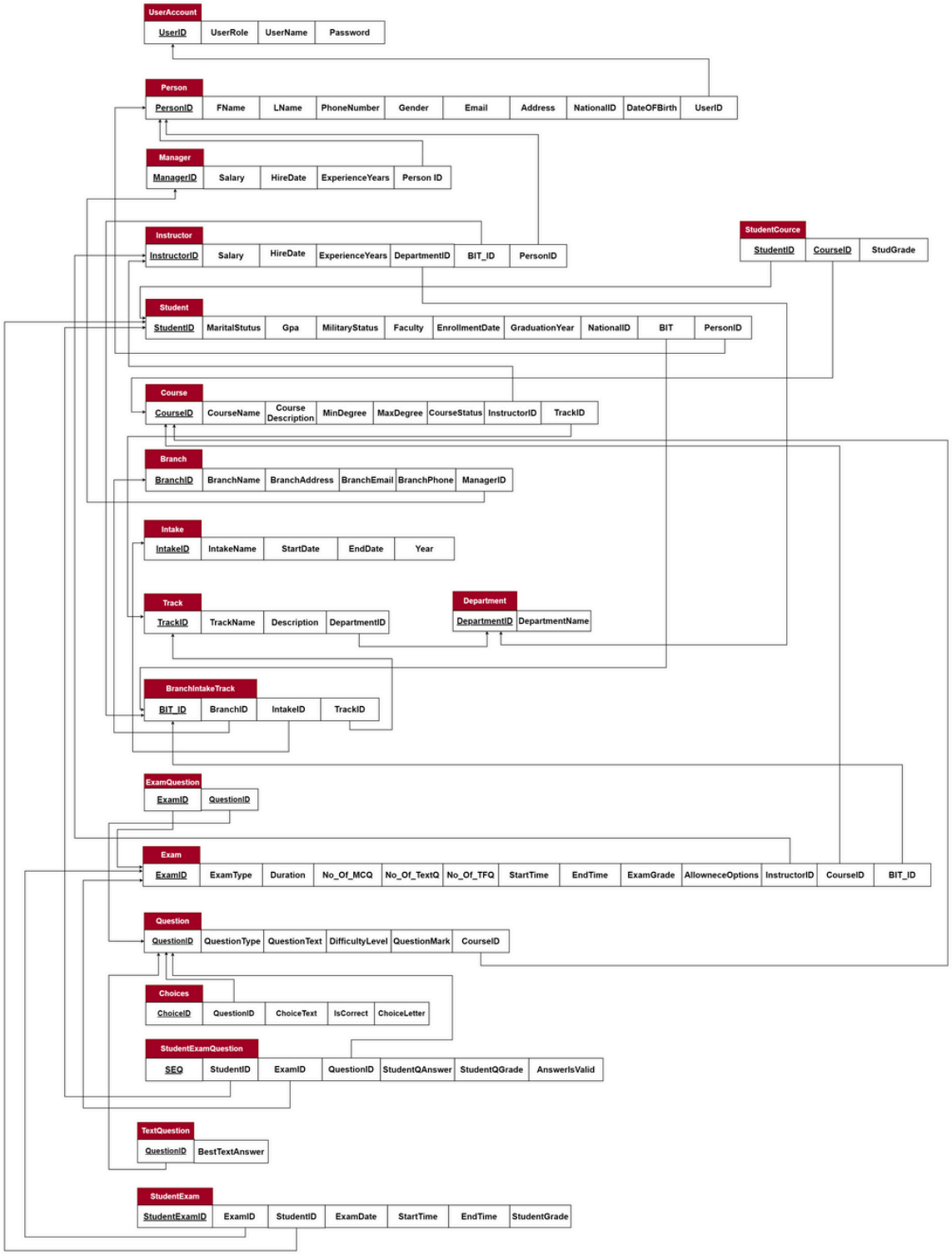
- **UserAccount – Person** (1:1)
- **Person – Manager** (1:1)
- **Person – Student** (1:1)
- **Person – Instructor** (1:1)
- **Manager – Branch** (1:1)
- **Department – Track** (1:N)
- **Branch – Intake – Track** (M:N)
- **BranchIntakeTrack – Student** (1:N)
- **BranchIntakeTrack – Instructor** (1:N)
- **Instructor – Course** (1:N)
- **Track – Course** (1:N)
- **Student – Course** (M:N)
- **Course – Exam** (1:N)
- **Exam – Question** (M:N)
- **Question – Choices** (1:N)
- **Question – TextQuestion** (1:1)
- **Student – Exam** (M:N)
- **Student – Exam – Question** (M:N)

4- Enhanced ERD

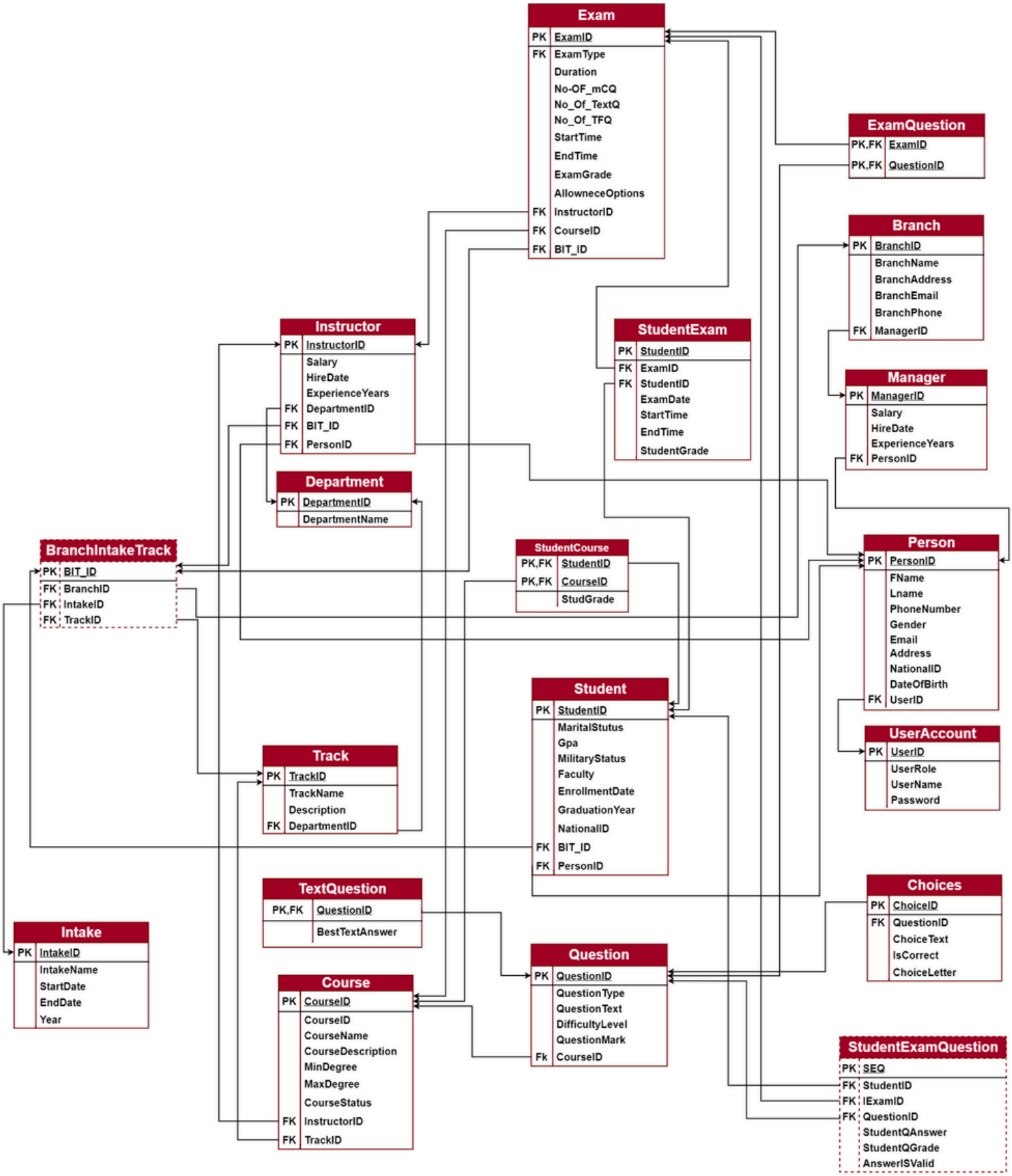


5- Mapping To Relational Tables

- Conceptual entities and relationships are converted into relationl database tables with defined primary keys (PK) and Foreign keys (FK)
- Each many to many relationship handled using **associative tables** such **StudentExamQuestion** , **BranchTrackIntake** , **StudentCourse ,etc**
- The strucure follows **normalization principles** to reduce redundency , enhance data organization and query effiecincy



3- Relational Schema



4-Database Storage Implementation Using Files and Filegroups in Sql Server

one of the project requirements says “implement your database in files and filegrou according to data size and your estimation” so to perform that on sql server you ne to perform the following steps :

- Files and file groups determination
- Assign each table to its own file group
- Size Estimation of each file group based on its the tables and implementation

1- Files and Filegroups Determination

The distribution of the database tables across fiegroups contain files allows parallel I/O operations across different disks if needed, improving database performance and scalability.here we divided the database into three filegroups in addition to the transaction log:

- **PRIMARY Filegroup :** Holds small and core tables (metadata, lookup tables, administrative data).
- **FG_LargeTables Filegroup :** Dedicated to large and frequently updated tables (Students, Courses, Exams, Questions, etc.).
- **FG_Index Filegroup :** Stores non-clustered indexes separately to improve performance and reduce I/O contention.
- **Transaction Log:** Stored in a separate file for recovery and auditing purposes.

2- Table-to-Filegroup Assignment

- **PRIMARY Filegroup:**
  UserAccount - Department - Manager - Branch - Intake - Track
- **FG_LargeTables Filegroup:**
  Person - BranchIntakeTrack - Student - Instructor - Course - StudentCourse Exam - Question - Choices - TextQuestion - ExamQuestion - StudentExamQuestion - StudentExam
- **FG_Index Filegroup**
  Example: Non-clustered indexes on Person(Fname,Lname), Student(GPA), Exam(CourseID), etc.

3- Size Estimation of Filegroups

The estimation of the Filegroups size is the base for estimationg the **total database size** beacause we deal with cloud databases, storage costs money. so estimating filegroup size helps you predict monthly costs.

Table Size (Mb): The total estimated size of the table in Megabytes, calculated using the formula: **((Row size * Expected No Of Rows)) / 1024^2**

Student			
Keys	Column Name	Data Type	Size (Bytes)
PK	StudentID	INT	4
	MaritalStatus	NVARCHAR(20)	42
	GPA	DECIMAL(4,2)	5
	MilitaryStatus	NVARCHAR(20)	42
	Faculty	NVARCHAR(100)	202
	EnrollmentDate	DATE	3
	GraduationYear	INT	4
FK	BIT_ID	INT	4
FK	PersonID	INT	4
Row Size in Bytes			310
Expected Number Of Rows in the table			3000
Table Size (Mb)=((Row size * Expected No Of Rows))/1024^2			0.886917114

- **Calculate the estimate size of each table in the same way**

FileGroup	Table Name	Table size
Primary	UserAccount	3
	Department	1
	Manager	3
	Branch	1
	Intake	1
	Track	1
FileGroup Estimated size		10

- "We allocated the **FG_Index** filegroup as **30%** of the estimated data size, which provides sufficient storage for all non-clustered indexes and ensures separation from the base data."

FileGroup	Table Name	Table size
FG_LargeTables	Person	5
	Student	4
	Instructor	3
	BranchIntakeTrack	5
	Course	3
	StudentCourse	5
	Exam	2
	Question	5
	Choices	5
	TextQuestion	5
	ExamQuestion	5
	StudentExamQuestion	8
	StudentExam	5
FileGroup Estimated size		60

FG_Index Filegroup Estimated size = .3 x (50 + 10) = 18

5- Database implementation in Sql server

```
-- ================================================================
-- Create the database with three filegroups
-- ================================================================
CREATE DATABASE ITI_Project
ON PRIMARY
(
    NAME = ITI_PrimaryData, -- Default filegroup for small or core tables
    FILENAME = 'C:\ExaminationSystemProject\DatabaseFiles\ITI_PrimaryData.mdf',
    SIZE = 20MB,
    MAXSIZE = 500MB,
    FILEGROWTH = 10MB
),
FILEGROUP FG_LargeTables --Stores large and frequently updated tables
(
    NAME = ITI_LargeTables,
    FILENAME = 'C:\ExaminationSystemProject\DatabaseFiles\ITI_LargeTables.ndf',
    SIZE = 60MB,
    MAXSIZE = 1GB,
    FILEGROWTH = 20MB
),
FILEGROUP FG_Index  --Store non-clustered indexes
(
    NAME = ITI_IndexData,
    FILENAME = 'C:\ExaminationSystemProject\DatabaseFiles\ITI_IndexData.ndf',
    SIZE = 20MB,
    MAXSIZE = 500MB,
    FILEGROWTH = 10MB
)
LOG ON  --Transaction log
(
    NAME = ITI_Log,
    FILENAME = 'C:\ExaminationSystemProject\DatabaseFiles\ITI_Log.ldf',
    SIZE = 20MB,
    MAXSIZE = 200MB,
    FILEGROWTH = 5MB
);
```

Tables Creation

```sql
-- ================================================================
-- Database: ITI_Project
-- Purpose : Training Management System Schema
-- ================================================================
USE ITI_Project;
GO
-- ================================================================
-- 1. UserAccount - Stores system login credentials
-- ================================================================
CREATE TABLE UserAccount (
    UserID INT PRIMARY KEY IDENTITY(1,1),    -- Auto increment
    UserRole NVARCHAR(50) NOT NULL,                  -- Role (Admin, Instructor, etc.)
    UserName NVARCHAR(100) NOT NULL,                 -- Login username
    Password NVARCHAR(100) NOT NULL,                -- Hashed password
    CONSTRAINT UQ_UserAccount_UserName UNIQUE (UserName)  -- Unique constraint
)ON[PRIMARY];
GO
-- ================================================================
-- 2. Person - Stores personal details for all users
-- ================================================================
CREATE TABLE Person (
    PersonID INT PRIMARY KEY,
    FName NVARCHAR(50),
    LName NVARCHAR(50),
    PhoneNumber NVARCHAR(20),
    Gender NVARCHAR(10),
    Email NVARCHAR(100),
    Address NVARCHAR(255),
    NationalID VARCHAR(14),
    DateOfBirth DATE,
    UserID INT UNIQUE,                                  -- One account per person
        FOREIGN KEY (UserID) REFERENCES UserAccount(UserID) ON DELETE CASCADE
)ON FG_LargeTables;
GO
-- ================================================================
-- 3. Department - Academic or administrative departments
-- ================================================================
CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName NVARCHAR(100)
)ON [PRIMARY];
GO
-- ================================================================
-- 4. Manager - Special type of person with managerial role
-- ================================================================
CREATE TABLE Manager (
    ManagerID INT PRIMARY KEY,
    Salary DECIMAL(10,2),
    HireDate DATE,
    ExperienceYears INT,
    PersonID INT UNIQUE,
    FOREIGN KEY (PersonID) REFERENCES Person(PersonID) ON DELETE CASCADE
)ON [PRIMARY];
GO
```

```sql
-- ================================================================
-- 5. Branch - Represents physical campus locations
-- ================================================================
CREATE TABLE Branch (
    BranchID INT PRIMARY KEY,
    BranchName NVARCHAR(100),
    BranchAddress NVARCHAR(255),
    BranchEmail NVARCHAR(100),
    BranchPhone NVARCHAR(20),
    BranchManagerID INT UNIQUE,                      -- One manager per branch
    FOREIGN KEY (BranchManagerID) REFERENCES Manager(ManagerID) ON DELETE SET NULL
)ON [PRIMARY];
GO
-- ================================================================
-- 6. Intake - Represents student intakes/batches
-- ================================================================
CREATE TABLE Intake (
    IntakeID INT PRIMARY KEY,
    IntakeName NVARCHAR(100),
    StartDate DATE,
    EndDate DATE,
    Year INT
)ON [PRIMARY];
GO
-- ================================================================
-- 7. Track - Represents specialization tracks
-- ================================================================
CREATE TABLE Track (
    TrackID INT PRIMARY KEY,
    TrackName NVARCHAR(50) NOT NULL,
    Description NVARCHAR(255),
    DepartmentID INT NOT NULL,
    CONSTRAINT FK_Track_Department FOREIGN KEY (DepartmentID)
        REFERENCES Department(DepartmentID)
);

-- ================================================================
-- 8. BranchIntakeTrack - Links branch, intake, and track
-- ================================================================
CREATE TABLE BranchIntakeTrack(
    BIT_ID INT PRIMARY KEY,
    BranchID INT,
    IntakeID INT,
    TrackID INT,
    UNIQUE (BranchID, IntakeID, TrackID),
    FOREIGN KEY (BranchID) REFERENCES Branch(BranchID) ON DELETE CASCADE,
    FOREIGN KEY (IntakeID) REFERENCES Intake(IntakeID) ON DELETE CASCADE,
    FOREIGN KEY (TrackID) REFERENCES Track(TrackID) ON DELETE CASCADE
)ON FG_LargeTables;
GO
```

```sql
-- ================================================================
-- 9. Student - Academic records for students
-- ================================================================
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    MaritalStatus NVARCHAR(20),
    GPA DECIMAL(4,2),
    MilitaryStatus NVARCHAR(20),
    Faculty NVARCHAR(100),
    EnrollmentDate DATE,
    GraduationYear INT,
    BIT_ID INT,
    PersonID INT UNIQUE,
    FOREIGN KEY (BIT_ID) REFERENCES BranchIntakeTrack(BIT_ID) ON DELETE CASCADE,
    FOREIGN KEY (PersonID) REFERENCES Person(PersonID) ON DELETE CASCADE
)ON FG_LargeTables;
GO
```

```sql
-- ================================================================
-- 10. Instructor - Teaching staff
-- ================================================================
CREATE TABLE Instructor (
    InstructorID INT PRIMARY KEY,
    Salary DECIMAL(10,2),
    HireDate DATE,
    ExperienceYears INT,
    DepartmentID INT,
    PersonID INT UNIQUE,
    BIT_ID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID) ON DELETE SET NULL,
    FOREIGN KEY (PersonID) REFERENCES Person(PersonID) ON DELETE CASCADE,
    FOREIGN KEY (BIT_ID) REFERENCES BranchIntakeTrack(BIT_ID) ON DELETE SET NULL
)ON FG_LargeTables;
-- ================================================================
-- 11. Course - Academic courses
-- ================================================================
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName NVARCHAR(50),
    CourseDescription NVARCHAR(MAX),
    MinDegree DECIMAL(6,2),
    MaxDegree DECIMAL(6,2),
    CourseStatus NVARCHAR(20) DEFAULT 'Active',
    InstructorID INT,
    TrackID INT,
    FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID) ON DELETE SET NULL,
    FOREIGN KEY (TrackID) REFERENCES Track(TrackID) ON DELETE SET NULL
)ON FG_LargeTables;
-- ================================================================
-- 12. StudentCourse - Links students to enrolled courses
-- ================================================================
CREATE TABLE StudentCourse (
    StudentID INT,
    CourseID INT,
    StudGrade DECIMAL(6,2),
    PRIMARY KEY (StudentID, CourseID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID) ON DELETE CASCADE,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID) ON DELETE CASCADE
)ON FG_LargeTables;
-- ================================================================
-- 13. Exam - Exams assigned for courses
-- ================================================================
CREATE TABLE Exam (
    ExamID INT PRIMARY KEY,
    ExamType NVARCHAR(50),
    BIT_ID INT,
    Duration INT,      -- Duration in minutes
    No_Of_MCQ INT,
    No_Of_TextQ INT,
    No_Of_TFQ INT,
    MaxGrade DECIMAL(6,2),
    MinGrade DECIMAL(6,2) NOT NULL DEFAULT 50.00,
    AllowanceOptions NVARCHAR(100),
    InstructorID INT,
    CourseID INT,
    FOREIGN KEY (BIT_ID) REFERENCES BranchIntakeTrack(BIT_ID) ON DELETE CASCADE,
    FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID) ON DELETE SET NULL,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID) ON DELETE CASCADE
```

```sql
-- ================================================================
-- 14. Question - Stores exam questions
-- ================================================================
CREATE TABLE Question (
    QuestionID INT IDENTITY(1,1) PRIMARY KEY,
    QuestionType NVARCHAR(50) NOT NULL,              -- MCQ, TF, Text
    QuestionText NVARCHAR(MAX) NOT NULL,
    DifficultyLevel VARCHAR(20) DEFAULT 'Medium',
    QuestionMark DECIMAL(6,2) NOT NULL,
    CourseID INT NOT NULL,
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID) ON DELETE CASCADE
)ON FG_LargeTables;
-- ================================================================
-- 15. Choices - Options for MCQ/TF questions
-- ================================================================
CREATE TABLE Choices (
    ChoiceID INT IDENTITY(1,1) PRIMARY KEY,
    QuestionID INT NOT NULL,
    ChoiceText NVARCHAR(255) NOT NULL,
    IsCorrect BIT NOT NULL DEFAULT 0,
        ChoiceLetter CHAR(1) NOT NULL,
    FOREIGN KEY (QuestionID) REFERENCES Question(QuestionID) ON DELETE CASCADE
)ON FG_LargeTables;
-- ================================================================
-- 16. TextQuestion - Model answers for text questions
-- ================================================================
CREATE TABLE TextQuestion (
    QuestionID INT PRIMARY KEY,
    BestTextAnswer NVARCHAR(MAX) NULL,
    FOREIGN KEY (QuestionID) REFERENCES Question(QuestionID) ON DELETE CASCADE
)ON FG_LargeTables;
-- ================================================================
-- 17. ExamQuestion - Links exams to their assigned questions
-- ================================================================
CREATE TABLE ExamQuestion (
    ExamID INT,
    QuestionID INT,
    PRIMARY KEY (ExamID, QuestionID),
    FOREIGN KEY (ExamID) REFERENCES Exam(ExamID),
    FOREIGN KEY (QuestionID) REFERENCES Question(QuestionID)
)ON FG_LargeTables;
-- ================================================================
-- 18. StudentExamQuestion - Stores student answers for each question
-- ================================================================
CREATE TABLE StudentExamQuestion(
    SEQ INT PRIMARY KEY,
    StudentID INT,
    ExamID INT,
    QuestionID INT,
    StudentQAnswer NVARCHAR(MAX),
        AnswerIsValid BIT DEFAULT NULL,   -- whether student's answer is valid/correct
    StudentQGrade DECIMAL(6,2) NULL, -- grade for the student on this question
    UNIQUE (StudentID, ExamID, QuestionID),
    FOREIGN KEY (QuestionID) REFERENCES Question(QuestionID) ON DELETE NO ACTION,
    FOREIGN KEY (ExamID) REFERENCES Exam(ExamID) ON DELETE NO ACTION,
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID) ON DELETE CASCADE
)ON FG_LargeTables
GO
```

```sql
--19. Table to store which students are assigned to which exam
-- ================================================================
CREATE TABLE StudentExam (
    StudentExamID INT PRIMARY KEY IDENTITY(1,1),
    StudentID INT NOT NULL,
    ExamID INT NOT NULL,
    ExamDate DATE NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    StudentGrade DECIMAL(5,2) NULL, -- percentage or score
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    FOREIGN KEY (ExamID) REFERENCES Exam(ExamID),
    CONSTRAINT UQ_Student_Exam UNIQUE (StudentID, ExamID) -- ensures one exam per student
)ON FG_LargeTables;
```

6- Admin Account Creation

First of all, we create an Admin account who will have full control over the examination system database.

```
-------------------------------------
-- 1) Create login on the server (if not exists)
-------------------------------------
If Not Exists (Select 1 From sys.server_principals Where name = 'Admin')
Begin
    Create Login Admin
    With Password = 'StrongPassword123';  -- Change password as needed
End
Go

-------------------------------------
-- 2) Map the login to ITI_Project database user
-------------------------------------
Use ITI_Project;
Go

If Not Exists (Select 1 From sys.database_principals Where name = 'Admin')
Begin
    Create User Admin For Login Admin;
End
Go

-------------------------------------
-- 3) Grant full permissions (db_owner role) for the admin
-------------------------------------
Alter Role db_owner Add Member Admin;
Go

-------------------------------------
-- 4) Add Admin account to UserAccount table
-- Assumes UserAccount has: UserName, Password, UserRole
-------------------------------------
Declare @Password NVARCHAR(100) = 'StrongPassword123';

If Not Exists (Select 1 From UserAccount Where UserName = 'Admin')
Begin
    Insert Into UserAccount (UserName, Password, UserRole)
    Values ('Admin', HashBytes('SHA2_256', @Password), 'Admin');
End
```

Admin permissions and related objects

Role	permissions and tasks
Admin	• Create and manage user accounts and assign roles (supermangr , BranchManager,Instructor, Student). • has full permissions on the database objects • Grant/Revoke permissions on tables, views, stored procedures, and functions. • Create/maintain stored procedures, functions, views, and triggers. • Manage backups and recovery plans. • Monitor database performance and security. • Responsible for the performence optimization of the database

```
--=============================================================================================
--========================== 1- Creating the four roles in the database system=================
--=============================================================================================
-- Drop roles if they already exist (safe re-run)
If Exists (Select * From sys.database_principals Where name = 'SuperManagerRole')
    Drop Role SuperManagerRole;
If Exists (Select * From sys.database_principals Where name = 'ManagerRole')
    Drop Role BranchManagerRole;
If Exists (Select * From sys.database_principals Where name = 'InstructorRole')
    Drop Role InstructorRole;
If Exists (Select * From sys.database_principals Where name = 'StudentRole')
    Drop Role StudentRole;
Go

-- Create roles
Create Role SuperManagerRole;
Create Role BranchManagerRole;
Create Role InstructorRole;
Create Role StudentRole;
Go
```

Object Name	Type	Description
addSuperManagerUser	Stored procedure	Add SuperManger to the system
addManagerUser	Stored procedure	Add BranchManger to the system
AddInstructorUser	Stored procedure	Add instructor to the system
addStudentUser	Stored procedure	Add student to the system

SuperManager Related Objects

Object Name	Type	Description
dbo.GetCurrentManagerID()	Function	Return the id of the current manger logged to the sql server and all other objects use it
updateManagerUser	SP	update manager details
dbo.addBranch	SP	add new ITI branch
dbo.updateBranch	SP	update ITI branch details
dbo.addIntake	SP	add new intake
dbo.updateIntake	SP	update intake details
dbo.addDepartment	SP	add new department
dbo.updateDepartment	SP	update department details
dbo.addTrack	SP	add new track
dbo.updateTrack	SP	updatetrack details
dbo.deleteTrack	SP	delete track

SuperManager Related Objects

Object Name	Type	Description
dbo.v_ManagerDetails	view	Shows all managers with personal info and assigned branch
dbo.SearchManagers	SP	SP based on the v_ManagerDetails view for Search managers dynamically by optional filters such as BranchID, ManagerID, or Name
dbo.v_StudentDetails	view	Shows all students in all branches with personal info
dbo.SearchStudents	SP	stored procedure based on the v_StudentDetails view used for Searching students dynamically by optional filters
dbo.v_InstructorDetails	view	Shows all instructors in all branches with personal info
dbo.SearchInstructors	SP	SP based on the v_InstructorDetails view used for Searching instructors dynamically by optional filters
dbo.v_AllBranchesDetails	view	Shows all branches with related department, track, and intake info
dbo.SearchBranches	SP	Search branches dynamically with optional filters

BranchManager Related Objects

Object Name	Type	Description
dbo.GetCurrentManagerID	Function	Returns the Branch Manager ID of the currently logged-in user in SQL Server.
dbo.AddTrackToIntake	SP	open track in the current mager branch
dbo.AddCourse	SP	Add cource to track
dbo.UpdateCourse	SP	update course details
dbo.DeleteCourse	SP	Delete course in a track
dbo.assignStudentToCourse	SP	Assign students to specific course
dbo.updateStudentUser	SP	update student in the current manger branch
dbo.deleteStudentUser	SP	delete student in the current manger branch
dbo.UpdateInstructorUser	SP	update instructor in the current manger branch
dbo.DeleteInstructorUser	SP	delete instructor in the current manger branch

BranchManager Related Objects

Object Name	Type	Description
dbo.v_ManagerDetails	view	Shows the personal info for the current branchManager
dbo.SearchManagers	SP	Searching managers dynamically by optional filters such as BranchID, ManagerID, or Name
dbo.v_StudentDetails	view	Shows the personal info for the students in the branch of the current branchManger
dbo.SearchStudents	SP	stored procedure based on the v_StudentDetails view used for Searching students dynamically by optional filters
dbo.v_InstructorDetails	view	Shows the personal info for the instructors in the branch of the current branchManger
dbo.SearchInstructors	SP	SP based on the v_InstructorDetails view used for Searching instructors dynamically by optional filters

Instructor Related Objects

Object Name	Type	Description
dbo.GetCurrentInstructorID	Function	Retrieve the logged-in instructor's ID That function used by most inst objects
dbo.ShowCurrentInstructor Courses	View	Show courses taught by the current instructor
dbo.sp_createManualExam	SP	Create an exam manually for a course the current instructor teaches
dbo.sp_createRandomExam	SP	Create a random exam for a course the current instructor teaches
dbo.sp_updateExam	SP	Update an exam related to the current instructor
dbo.sp_deleteExam	SP	Delete an exam related to the current instructor
dbo.sp_assignExamToCourse Students	SP	Assign exam to course students
dbo.sp_addMCQQuestion	SP	Add Multiple Choice Question to course questions pool
dbo.sp_addTFQuestion	SP	Add True/False Question
dbo.sp_addTextQuestion	SP	Add True/False Question
dbo.sp_updateQuestion	SP	Update an existing question
dbo.sp_deleteQuestion	SP	Delete a question

Instructor Related Objects

Object Name	Type	Description
dbo.sp_viewInstructor Questions	Function	View all questions created by the current instructor with different options
dbo.v_InstructorDetails	view	Shows the current instructors personal info
dbo.sp_ViewStudentAnswers	SP	View students' answers in a specific exam
dbo.sp_CorrectExamManually	SP	Manually correct student answers on text questions
dbo.sp_UpdateExamTotalGrade	SP	Update Student total grade on an an exam after being all questions are corrected
dbo.invalid_question_tr	Trigger	instractor don't put invalid question

Student Related Objects

Object Name	Type	Description
dbo.GetCurrentStudentID	Function	Returns the id of the curre
dbo.vw_CurrentStudentExam Questions	SP	View students' answers in a specific exam
dbo.v_InstructorDetails	view	Shows the current instructors personal info
ExamTimePassed_tr	Trigger	prevent student from solving exam's question after the defined time

7- permissions Assignment To Each Role By The Admin

```
-- Admin has all permissions on the database

-- ============================================================================
--========================== 2- SuperManager Permissions=======================
============================================================================

--================================CORE FUNCTION==================================*/
Grant Execute On dbo.GetCurrentManagerID To SuperManagerRole;
/*============================================================================
  VIEWS: SuperManager can Select; BranchManager cannot Select directly
============================================================================*/
Grant Select On dbo.v_StudentDetails      To SuperManagerRole;
Grant Select On dbo.v_ManagerDetails      To SuperManagerRole;
Grant Select On dbo.v_InstructorDetails   To SuperManagerRole;
Grant Select On dbo.v_AllBranchesDetails  To SuperManagerRole;
/*============================================================================
  PROCEDURES — SuperManager only
============================================================================*/
Grant Execute On dbo.updateManagerUser    To SuperManagerRole;
Grant Execute On dbo.addBranch            To SuperManagerRole;
Grant Execute On dbo.updateBranch         To SuperManagerRole;
Grant Execute On dbo.addIntake            To SuperManagerRole;
Grant Execute On dbo.updateIntake         To SuperManagerRole;
Grant Execute On dbo.addDepartment        To SuperManagerRole;
Grant Execute On dbo.updateDepartment     To SuperManagerRole;
Grant Execute On dbo.addTrack             To SuperManagerRole;
Grant Execute On dbo.updateTrack          To SuperManagerRole;
Grant Execute On dbo.deleteTrack          To SuperManagerRole;
/*============================================================================
  SEARCH/REPORTING PROCEDURES — both roles can Execute
============================================================================*/
Grant Execute On dbo.SearchStudents     To SuperManagerRole , BranchManagerRole;
Grant Execute On dbo.SearchManagers     To SuperManagerRole ,BranchManagerRole;
Grant Execute On dbo.SearchInstructors  To SuperManagerRole ,BranchManagerRole;
Grant Execute On dbo.SearchBranches     To SuperManagerRole ,BranchManagerRole;
GRANT SELECT ON dbo.Manager TO SuperManagerRole;
GRANT SELECT ON dbo.Branch TO SuperManagerRole;
GRANT SELECT ON dbo.Intake TO SuperManagerRole;
GRANT SELECT ON dbo.Department TO SuperManagerRole;
GRANT SELECT ON dbo.Track TO SuperManagerRole;
-- ============================================================================
--========================== Manager Permissions==============================
/*============================================================================*/
GRANT EXECUTE ON dbo.GetCurrentManagerID TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.AddTrackToIntake TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.updateStudentUser  TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.deleteStudentUser  TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.UpdateInstructorUser  TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.DeleteInstructorUser  TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.AddCourse    TO SuperManagerRole, BranchManagerRole;
GRANT EXECUTE ON dbo.UpdateCourse TO SuperManagerRole, BranchManagerRole;
```

permissions Assignment To Each Role By The Admin

```
-- ================================================================================
--========================== Instructor Permissions=====================
-- ================================================================================
-- Allow Instructor to use the helper function
GRANT EXECUTE ON dbo.GetCurrentInstructorID        TO InstructorRole;
-- Allow Instructor to execute Exam-related procedures
GRANT EXECUTE ON dbo.sp_createRandomExam           TO InstructorRole;
GRANT EXECUTE ON dbo.sp_createManualExam           TO InstructorRole;
GRANT EXECUTE ON dbo.SP_ShowCourceExams            TO InstructorRole;
GRANT EXECUTE ON dbo.sp_updateExam                 TO InstructorRole;
GRANT EXECUTE ON dbo.sp_deleteExam                 TO InstructorRole;
GRANT EXECUTE ON dbo.sp_assignExamToCourseStudents TO InstructorRole;
-- Allow Instructor to execute Question-related procedures
GRANT EXECUTE ON dbo.sp_addMCQQuestion             TO InstructorRole;
GRANT EXECUTE ON dbo.sp_addTFQuestion              TO InstructorRole;
GRANT EXECUTE ON dbo.sp_addTextQuestion            TO InstructorRole;
GRANT EXECUTE ON dbo.sp_updateQuestion             TO InstructorRole;
GRANT EXECUTE ON dbo.sp_deleteQuestion             TO InstructorRole;
GRANT EXECUTE ON dbo.sp_viewInstructorQuestions    TO InstructorRole;
-- Allow Instructor to execute Student-Answer related procedures
GRANT EXECUTE ON dbo.sp_ViewStudentAnswers         TO InstructorRole;
GRANT EXECUTE ON dbo.sp_CorrectExamManually        TO InstructorRole;
GRANT EXECUTE ON dbo.sp_UpdateExamTotalGrade       TO InstructorRole;

-- Allow Instructor to view their courses
GRANT EXECUTE ON dbo.ShowCurrentInstructorCourses  TO InstructorRole;
-- ================================================================================
--========================== Student Permissions=========================
-- ================================================================================
-- Grant basic SELECT on Views
GRANT SELECT ON dbo.vw_CurrentStudentExamQuestions TO StudentRole;
-- Grant EXECUTE on Procedures & Functions
GRANT EXECUTE ON dbo.sp_StudentAnswerQuestion TO StudentRole;
GRANT EXECUTE ON dbo.fn_CompareTextAnswer TO StudentRole;
GRANT EXECUTE ON dbo.GetCurrentStudentID TO StudentRole;
```

8- Performance Optimization

This section describes the strategies used to optimize the performance of the Examination System Database to ensure scalability, fast response times, and efficient resource usage.

1. Indexing Strategy

Clustered Indexes:

Created by default On all primary keys (UserID,PersonID, ManagerID,InstructorID, StudentID, CourseID, ExamID, QuestionID, etc.).

NonClustered Indexes:

Nonclustered indexes were added to improve query performance and support frequent filtering and join operations in the database. making specific queries faster without altering the underlying data storage.

Index Name	Table Name	Type	Description
IX_Student_PersonID	Student	PersonID	Improves join performance between Student and Person using the foreign key.
IX_Student_FullName	Person	FName, LName	Optimizes searches filtering by full name (first + last name).
IX_Branch_BranchName	Branch	BranchName	Speeds up queries filtering by branch name

9- End-to-End Exam Process

In this section, we will create a full scenario of how our examination system works from the perspective of different roles. The workflow covers the complete lifecycle of the exam process, starting with instructor login and question management, moving through exam creation and student participation, and ending with evaluation and result generation.

Instructor Login

1- instructor login and execute the following SP to show his cources

```
EXEC ShowCurrentInstructorCourses
```

	CourseID	CourseName	CourseDescription	MinDegree	MaxDegree	CourseStatus	TrackID
1	103	Linux	Linux administration and shell basics	50.00	100.00	Inactive	1
2	102	Python	Python programming fundamentals	50.00	100.00	Active	1
3	101	SQL	DATABASE QUERY language course	50.00	100.00	Active	1

2- instructor add questions to the of different types to question pool such the following sample

```
EXEC sp_addMCQQuestion
@QuestionText = 'Which SQL command is used to remove a table permanently?',@CourseID = 101,
@Choice1 = 'DELETE TABLE table_name;',
@Choice2 = 'DROP TABLE table_name;',
@Choice3 = 'REMOVE TABLE table_name;',@Choice4 = 'TRUNCATE TABLE table_name;',@CorrectChoice = 'B';

EXEC sp_addTFQuestion @QuestionText = 'SQL is case-insensitive for keywords.', @CourseID = 101,
@CorrectChoice = 'A';  -- True

EXEC sp_addTextQuestion @QuestionText = 'Describe the ACID properties in database transactions.',
@CourseID = 101,
@BestTextAnswer = 'ACID stands for Atomicity, Consistency, Isolation, Durability which ensures reliable
 transactions.';
```

```
Exec sp_viewInstructorQuestions @CourseID = 101
```

QuestionID	CourseID	CourseName	QuestionType	QuestionText	DifficultyLevel	QuestionMark	ChoiceText	IsCorrect
1	101	SQL	MCQ	Which SQL command is used to remove a table perm...	Hard	10.00	DELETE TABLE table_name;	0
1	101	SQL	MCQ	Which SQL command is used to remove a table perm...	Hard	10.00	DROP TABLE table_name;	1
1	101	SQL	MCQ	Which SQL command is used to remove a table perm...	Hard	10.00	REMOVE TABLE table_na...	0
1	101	SQL	MCQ	Which SQL command is used to remove a table perm...	Hard	10.00	TRUNCATE TABLE table_n...	0

QuestionID	CourseID	CourseName	QuestionType	QuestionText	DifficultyLevel	QuestionMark	ChoiceText	IsCorrect
14	101	SQL	TF	SQL is case-insensitive for keywords.	Medium	5.00	True	1
14	101	SQL	TF	SQL is case-insensitive for keywords.	Medium	5.00	False	0

QuestionID	CourseID	CourseName	QuestionType	QuestionText	DifficultyLevel	QuestionMark	ChoiceText	IsCorrect
22	101	SQL	Text	Describe the ACID properties in database transactions.	Medium	5.00	NULL	NULL

Exams creation

```
EXEC sp_createRandomExam  @ExamID = 1001, @ExamType = 'Exam', @BIT_ID = 7, @Duration = 60,
@No_Of_MCQ = 5, @No_Of_TextQ =3,@No_Of_TFQ = 2,
@MaxGrade = 100, @AllowanceOptions = 'None',@CourseID = 101,@MinGrade = 50;
```

```
EXEC sp_createRandomExam  @ExamID = 1002, @ExamType = 'Exam',@BIT_ID = 7,
@Duration = 12 ,@No_Of_MCQ = 8,
@No_Of_TextQ =4, @No_Of_TFQ = 3,
@MaxGrade = 150,@AllowanceOptions = 'Calculator Allowed',
@CourseID = 101, @MinGrade = 60;
```

```
EXEC sp_createManualExam
    @ExamID = 1003, @ExamType = 'Exam',@BIT_ID = 7,
    @Duration = 90,@MaxGrade = 100,@AllowanceOptions = 'No Calculator',
    @CourseID = 101,@MinGrade = 50,@No_Of_MCQ = 3,@No_Of_TextQ = 2,@No_Of_TFQ = 2,
    @QuestionIDs = '2,3,4,5,6,7,8';  -- Comma-separated list of QuestionIDs
```

```
Exec SP_ShowCourceExams 101
```

	ExamID	CourseID	CourseName	ExamType	MinGrade	MaxGrade	No_Of_MCQ	No_Of_TextQ	No_Of_TFQ
1	1001	101	SQL	Exam	50.00	100.00	5	3	2
2	1002	101	SQL	Exam	60.00	150.00	8	4	3
3	1003	101	SQL	Exam	50.00	100.00	3	2	2

```
EXEC sp_assignExamToCourseStudents
    @ExamID    = 1001,
    @CourseID = 101,
    @ExamDate = '2025-12-15',
    @StartTime = '09:00:00',
    @EndTime   = '11:00:00';
```

Student Login

```
select * from  dbo.vw_CurrentStudentExamQuestions
```

	QuestionID	QuestionText	QuestionType	QuestionMark	Choices
1	2	Which keyword is used to sort results in SQL?	MCQ	5.00	ORDER BY | SORT BY | GROUP BY | HAVING
2	6	Which SQL command is used to modify table struct...	MCQ	5.00	ALTER TABLE | UPDATE TABLE | MODIFY TABLE | CHAN...
3	7	Which operator is used for pattern matching in SQL?	MCQ	5.00	LIKE | MATCH | PATTERN | REGEX
4	8	Which constraint ensures unique values in a column?	MCQ	5.00	PRIMARY KEY | FOREIGN KEY | UNIQUE | CHECK
5	9	Which SQL function returns the current system date?	MCQ	5.00	NOW() | GETDATE() | CURRENT_DATE | SYSDATE()
6	16	INNER JOIN returns only matching rows between ta...	TF	5.00	True | False
7	20	A table can have multiple primary keys.	TF	5.00	True | False
8	22	Describe the ACID properties in database transacti...	Text	5.00	NULL
9	24	Explain the difference between a primary key and a ...	Text	5.00	NULL
10	27	What is a stored procedure and its benefits?	Text	5.00	NULL

```
Exec sp_StudentAnswerQuestion 2 , 'B'
Exec sp_StudentAnswerQuestion 6 ,'A'
Exec sp_StudentAnswerQuestion 7 ,'A'
Exec sp_StudentAnswerQuestion 8 ,'A'
Exec sp_StudentAnswerQuestion 9 , 'A'
Exec sp_StudentAnswerQuestion 16 , 'A'
Exec sp_StudentAnswerQuestion 20 , 'A'
Exec sp_StudentAnswerQuestion 22 , 'ACID stands for Atomicity, Consistency, Isolation, Durability'
Exec sp_StudentAnswerQuestion 24 , 'Primary key uniquely identifies rows and cannot be null'
Exec sp_StudentAnswerQuestion 27 , 'stored procedure is a precomiled'
```

- **Once the student assigned his answers , the TF and MCQ questions are corrected automatically and assigned the grade**

Exams Correction

Instructor Login

```
EXEC sp_ViewStudentAnswers @StudentID = 6, @ExamID = 1001
```

SEQ	StudentID	StudentFullName	ExamID	ExamType	CourseID	CourseName	QuestionID	QuestionText	QuestionType	ModelAnswer	StudentQAnswer	AnswerIsValid	StudentQGrade
1	6	Ahmed Alaa	1001	Exam	101	SQL	2	Which keyword is used t...	MCQ	NULL	B	0	0.00
2	6	Ahmed Alaa	1001	Exam	101	SQL	6	Which SQL command is...	MCQ	NULL	A	1	5.00
3	6	Ahmed Alaa	1001	Exam	101	SQL	7	Which operator is used f...	MCQ	NULL	A	1	5.00
4	6	Ahmed Alaa	1001	Exam	101	SQL	8	Which constraint ensure...	MCQ	NULL	A	0	0.00
5	6	Ahmed Alaa	1001	Exam	101	SQL	9	Which SQL function retu...	MCQ	NULL	A	0	0.00
6	6	Ahmed Alaa	1001	Exam	101	SQL	16	INNER JOIN returns only...	TF	NULL	A	1	5.00
7	6	Ahmed Alaa	1001	Exam	101	SQL	20	A table can have multipl...	TF	NULL	A	0	0.00
8	6	Ahmed Alaa	1001	Exam	101	SQL	22	Describe the ACID prop...	Text	ACID stands fo...	ACID stands for ...	1	NULL
9	6	Ahmed Alaa	1001	Exam	101	SQL	24	Explain the difference be...	Text	Primary key uni...	Primary key uni...	1	NULL
10	6	Ahmed Alaa	1001	Exam	101	SQL	27	What is a stored proced...	Text	A stored proce...	stored procedur...	1	NULL

```
EXEC sp_ViewStudentAnswers @StudentID = 6, @CourseID = 101, @QuestionType = 'Text';
```

SEQ	StudentID	StudentFullName	ExamID	ExamType	CourseID	CourseName	QuestionID	QuestionText
8	6	Ahmed Alaa	1001	Exam	101	SQL	22	Describe the ACID properties in database transa...
9	6	Ahmed Alaa	1001	Exam	101	SQL	24	Explain the difference between a primary key and...
10	6	Ahmed Alaa	1001	Exam	101	SQL	27	What is a stored procedure and its benefits?

- **Now the instructor corrects the text questions which are not corrected automatically**

```
EXEC sp_CorrectExamManually @StudentID = 6, @ExamID = 1001, @QuestionID = 22, @Grade = 8.00;
EXEC sp_CorrectExamManually @StudentID = 6, @ExamID = 1001, @QuestionID = 24, @Grade = 8.00;
EXEC sp_CorrectExamManually @StudentID = 6, @ExamID = 1001, @QuestionID = 27, @Grade = 8.00;
```

```
EXEC sp_UpdateExamTotalGrade @StudentID = 6 , @ExamID = 1001
```

	StudentExamID	StudentID	ExamID	ExamDate	StartTime	EndTime	StudentGrade
1	1	6	1001	2025-08-15	09:00:00.0000000	11:00:00.0000000	60.00
2	2	7	1001	2025-12-15	09:00:00.0000000	11:00:00.0000000	NULL
3	3	8	1001	2025-12-15	09:00:00.0000000	11:00:00.0000000	NULL
4	4	9	1001	2025-12-15	09:00:00.0000000	11:00:00.0000000	NULL
5	5	10	1001	2025-12-15	09:00:00.0000000	11:00:00.0000000	NULL

10- Full Database Backup Using SQL Server Agent

This job automates the process of creating a full backup of the ITI_Project database at scheduled times. It ensures data safety and recovery by keeping timestamped backup files.