Student: Ridvan Plluzhina

Subject: Introduction to databases

Student number: 21286

Content:

a) Structured and organized requirements

We want to store data about students enrolled in our institution for certain exams. For each student, we want to remember their StudentID,first name, last name, date of birth, email (optional), and their level of study (Bachelor's, Master's, or PhD). Some students might also be employed, and for these students, we want to store information about their work.
Each student can take multiple exams. For each exam a student takes, we want to record the ExamID and date of the exam. The exam is associated with a specific subject and organized by a professor. The exam is also held in a specific classroom.
For each exam, we want to know the subject it covers. Each subject has a unique SubjectID and name and is taught by a professor. We also need to manage classroom information, such as its ClassroomID, room number, building, and capacity.
We also need to keep track of the professors who organize and teach various subjects. For each professor, we want to remember their ProfessorID, first name, last name, email and phoneNumbers. For UniversityProfessors we want to store the Department (e.g Computer Science…) and for ExternalProfessor we want to store their Affiliation (e.g ABC Institute…). Professors can also mentor students, and we want to keep track of these mentorship relationships.
Additionally, we want to manage the information about classrooms. Each classroom can host multiple subjects, and we want to record which subjects are taught in each classroom.
Professors can mentor students, providing guidance and support.

Entities and Attributes:

Student:
StudentID (Primary Key), FirstName, LastName, DateOfBirth, Email (Optional), Level (Bachelor's, Master's, PhD)

EmployedStudent (ISA Student):
Work (Optional)

Exam:
ExamID (Primary Key), ExamDate

Subject:
SubjectID (Primary Key), SubjectName

Classroom:
ClassroomID (Primary Key), RoomNr, Building, Capacity

Professor:
ProfessorID (Primary Key), FirstName, LastName, Email, PhoneNumbers (Multivalued)

UniversityProfessor (ISA Professor):
Department

ExternalProfessor (ISA Professor):
Affiliation

Relationships:

Takes: Between Student and Exam

Covers: Between Exam and Subject

HeldIn: Between Exam and Classroom

Hosts: Between Classroom and Subject

TaughtBy: Between Subject and Professor

Mentors: Between Professor and Student

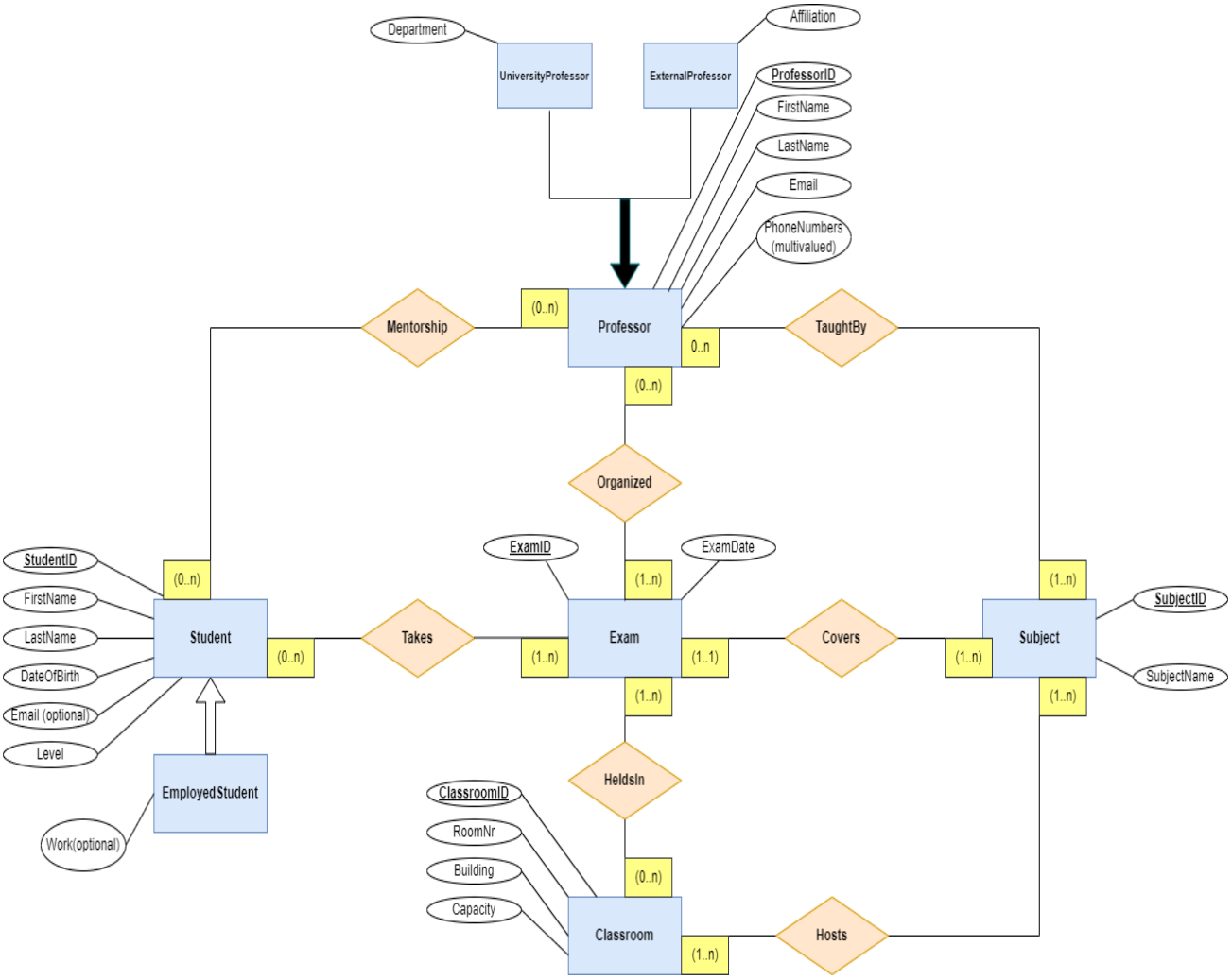
b) Glossary of Terms

Glossary of terms provides definitions for specialized or technical words and phrases used within a particular field, subject, or document. It is designed to help readers understand the terminology and concepts that might be unfamiliar to them.

GlossaryTerms

Term	Description	Synonyms	Connections
Student	An individual enrolled in the institution.	Learner	EmployedStudent, Takes, Mentors
EmployedStudent	A student who is also employed.	Working Student	Student
Exam	An assessment associated with a specific subject.	Test, Assessment	Takes, Covers, HeldIn
Subject	An academic course taught by a professor.	Course, Class	Covers, Hosts, TaughtBy
Classroom	A physical room where subjects are taught and exams are held.	Room, Lecture Hall	HeldIn, Hosts
Professor	An individual who teaches subjects and may mentor students.	Instructor, Teacher	TaughtBy, Mentors, UniversityProfessor, ExternalProfessor
UniversityProfessor	A professor affiliated with a department within the institution.	Internal Professor	Professor
ExternalProfessor	A professor affiliated with an external organization.	Visiting Professor	Professor

c) Diagram of the conceptual schema



d) Data dictionary of the conceptual schema

A data dictionary of the conceptual schema is a detailed description of the data structures, relationships, and constraints within a database, providing a blueprint for understanding the database's organization and the relationships between different data elements.

EntityDataDictionary

Entity	Description	Attributes
Student	An individual enrolled in the institution.	StudentID, FirstName, LastName, DateOfBirth, Email (Optional), Level (Bachelors, Masters, PhD)
EmployedStudent	A student who is also employed.	StudentID, Work (Optional)
Exam	An assessment associated with a specific subject.	ExamID, ExamDate
Subject	An academic course taught by a professor.	SubjectID, SubjectName
Classroom	A physical room where subjects are taught and exams are held.	ClassroomID, RoomNr, Building, Capacity
Professor	An individual who teaches subjects and may mentor students.	ProfessorID, FirstName, LastName, Email, PhoneNumbers (Multivalued)
UniversityProfessor	A professor affiliated with a department within the institution.	ProfessorID, Department
ExternalProfessor	A professor affiliated with an external organization.	ProfessorID, Affiliation

RelationshipDataDictionary

Relationship	Description	Components
Takes	Relationship indicating a student taking an exam.	Student, Exam
Covers	Relationship indicating an exam covering a subject.	Exam, Subject
HeldIn	Relationship indicating the classroom where an exam is held.	Exam, Classroom
Hosts	Relationship indicating the classroom where a subject is taught.	Classroom, Subject
TaughtBy	Relationship indicating the professor teaching a subject.	Subject, Professor
Mentors	Relationship indicating a professor mentoring a student.	Professor, Student

ExternalConstraintsDataDictionary

ConstraintDescription
A professor can only organize exams taught by himself.
No two exams should be held in the same classroom at the same time.

e) Table of volumes and table of operations

The Table of Volumes is a detailed estimate of the number of records for each entity and relationship in the database. It provides an approximation of how much data each table will hold, which helps in understanding the scale of the database and planning for storage, performance, and optimization needs.

Volumes

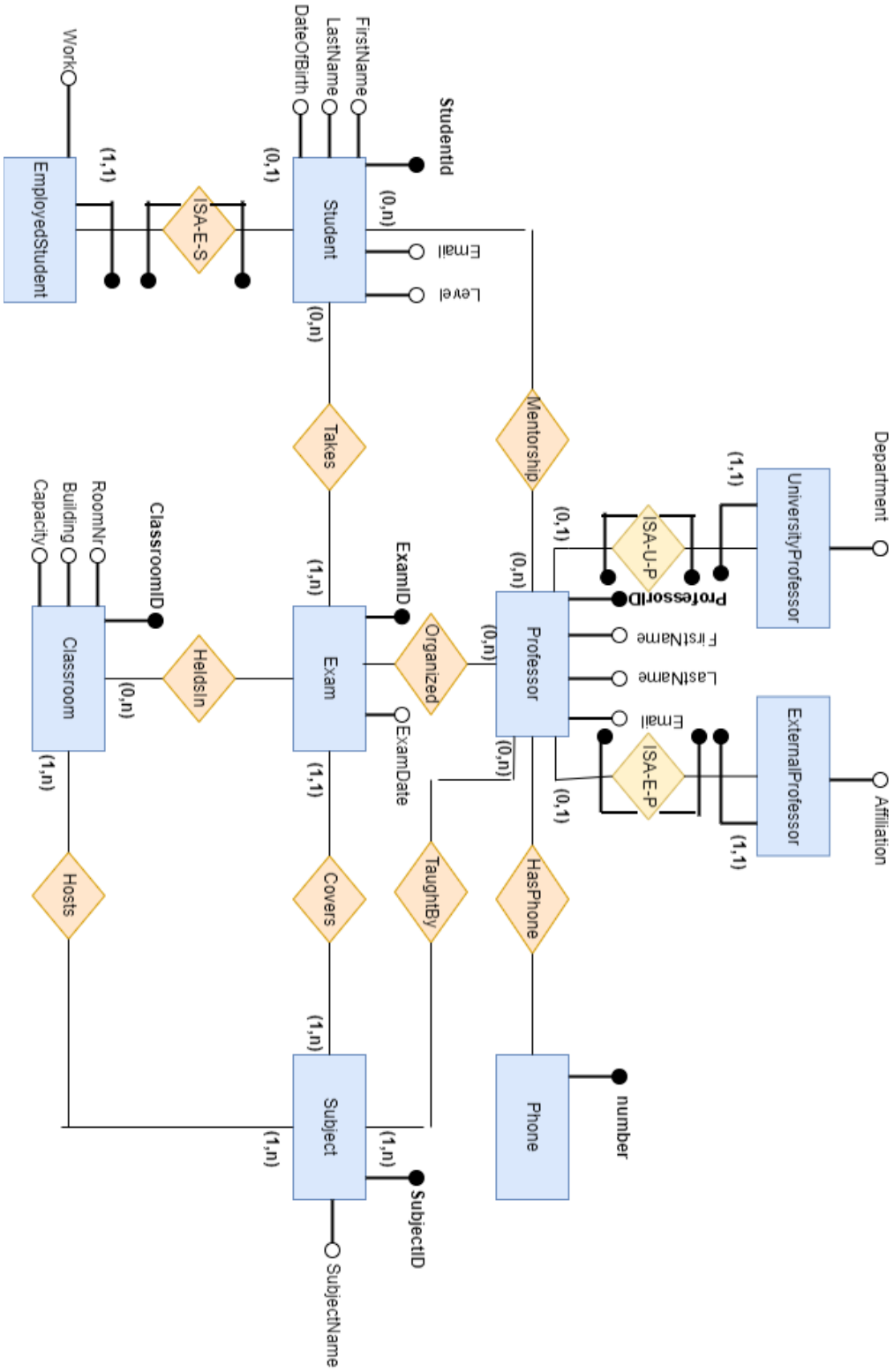
EntityOrRelationship	EstimatedVolume
Student	1000
EmployedStudent	300
Exam	500
Subject	50
Classroom	20
Professor	100
UniversityProfessor	50
ExternalProfessor	50
Takes	2000
Covers	500
HeldIn	500
Hosts	100
TaughtBy	50
Mentors	200

The Table of Operations lists the common database operations or queries that are expected to be performed, along with their expected frequency. This helps in understanding the workload on the database and is useful for optimizing performance and planning indexing strategies.

Operations

OperationDescription	Frequency
Add a new student	50
Add a new employed student	20
Schedule a new exam	100
Assign a subject to a professor	30
Assign a classroom to an exam	100
Register a student for an exam	200
Get the list of students for a given exam	150
Get the list of exams for a given student	150
Get the list of subjects taught by a professor	100
Get the list of classrooms hosting a subject	50
Add a new professor	10
Mentor assignment to a student by a professor	40

2) Restructuring of the conceptual schema



a) Steps for Restructuring

I performed a **redundancy analysis** to ensure efficiency and eliminate any redundant data. I eliminated **multi-valued** attributes by creating separate entities and relationships, such as introducing the Phone entity for professor phone numbers. I also removed any **composite attributes**, ensuring that all attributes are atomic. I addressed **ISA hierarchies and generalizations** by adding constraints for attributes shared by child entities of the same parent entity and ensuring disjointness and completeness. I chose **primary identifiers** for all entities, such as ProfessorID for the Professor entity, ensuring each entity has a unique identifier. I specified **external constraints**, including the unique exam schedule constraint (no two exams should be held in the same classroom at the same time) and the professor exam organization constraint (a professor can only organize exams for subjects they teach).Finally, I **reformulated the operations** and application load specifications to align with the restructured schema, detailing access types and frequencies in the access tables for each operation.

b) Cost of Evaluation

Cost of Evaluation refers to the analysis of database operations to understand their impact on performance. It involves determining how frequently different entities and relationships are accessed, the complexity of these accesses, and the types of operations performed. This helps in optimizing the database design for better performance.

AccessTableOperation1

Concept	Construct	Accesses	Type
Student	Entity	1	Read
Takes	Relationship	1	Read
Exam	Entity	1	Read

AccessTableOperation2

Concept	Construct	Accesses	Type
Exam	Entity	1	Read
Covers	Relationship	1	Read
Subject	Entity	1	Read

AccessTableOperation3

Concept	Construct	Accesses	Type
Professor	Entity	1	Read
Organized	Relationship	1	Read
Exam	Entity	1	Read

AccessTableOperation4

Concept	Construct	Accesses	Type
Exam	Entity	1	Read
HeldIn	Relationship	1	Write
Classroom	Entity	1	Read

AccessTableOperation5

Concept	Construct	Accesses	Type
Professor	Entity	1	Read
Mentorship	Relationship	1	Write
Student	Entity	1	Read

3) Direct translation to the relational model

Student(StudentID, FirstName, LastName, DateOfBirth, Email, Level)
EmployedStudent(StudentID, Work)
foreign key: EmployedStudent(StudentID) ⊆ Student(StudentID)
Professor(ProfessorID, FirstName, LastName, Email)
UniversityProfessor(ProfessorID, Department)
foreign key: UniversityProfessor(ProfessorID) ⊆ Professor(ProfessorID)
ExternalProfessor(ProfessorID, Affiliation)
foreign key: ExternalProfessor(ProfessorID) ⊆ Professor(ProfessorID)
Phone(Number, ProfessorID)
foreign key: Phone(ProfessorID) ⊆ Professor(ProfessorID)
Exam(ExamID, ExamDate, ProfessorID, SubjectID, ClassroomID)
foreign key: Exam(ProfessorID) ⊆ Professor(ProfessorID)
foreign key: Exam(SubjectID) ⊆ Subject(SubjectID)
foreign key: Exam(ClassroomID) ⊆ Classroom(ClassroomID)
Subject(SubjectID, SubjectName)
Classroom(ClassroomID, RoomNr, Building, Capacity)
Takes(StudentID, ExamID)
foreign key: Takes(StudentID) ⊆ Student(StudentID)
foreign key: Takes(ExamID) ⊆ Exam(ExamID)
Covers(ExamID, SubjectID)
foreign key: Covers(ExamID) ⊆ Exam(ExamID)
foreign key: Covers(SubjectID) ⊆ Subject(SubjectID)
HeldIn(ExamID, ClassroomID)
foreign key: HeldIn(ExamID) ⊆ Exam(ExamID)
foreign key: HeldIn(ClassroomID) ⊆ Classroom(ClassroomID)
Hosts(ClassroomID, SubjectID)
foreign key: Hosts(ClassroomID) ⊆ Classroom(ClassroomID)
foreign key: Hosts(SubjectID) ⊆ Subject(SubjectID)
TaughtBy(SubjectID, ProfessorID)
foreign key: TaughtBy(SubjectID) ⊆ Subject(SubjectID)
foreign key: TaughtBy(ProfessorID) ⊆ Professor(ProfessorID)
Mentors(ProfessorID, StudentID)
foreign key: Mentors(ProfessorID) ⊆ Professor(ProfessorID)
foreign key: Mentors(StudentID) ⊆ Student(StudentID)

4) Restructuring of the relational schema

Student(StudentID, FirstName, LastName, DateOfBirth, Email, Level, Work)
Professor(ProfessorID, FirstName, LastName, Email, Department, Affiliation)
Phone(Number, ProfessorID)
foreign key: Phone(ProfessorID) ⊆ Professor(ProfessorID)
Exam(ExamID, ExamDate, ProfessorID, SubjectID, ClassroomID)
foreign key: Exam(ProfessorID) ⊆ Professor(ProfessorID)
foreign key: Exam(SubjectID) ⊆ Subject(SubjectID)
foreign key: Exam(ClassroomID) ⊆ Classroom(ClassroomID)
Subject(SubjectID, SubjectName)
Classroom(ClassroomID, RoomNr, Building, Capacity)
Takes(StudentID, ExamID)
foreign key: Takes(StudentID) ⊆ Student(StudentID)
foreign key: Takes(ExamID) ⊆ Exam(ExamID)
Covers(ExamID, SubjectID)
foreign key: Covers(ExamID) ⊆ Exam(ExamID)
foreign key: Covers(SubjectID) ⊆ Subject(SubjectID)
HeldIn(ExamID, ClassroomID)
foreign key: HeldIn(ExamID) ⊆ Exam(ExamID)
foreign key: HeldIn(ClassroomID) ⊆ Classroom(ClassroomID)
Hosts(ClassroomID, SubjectID)
foreign key: Hosts(ClassroomID) ⊆ Classroom(ClassroomID)
foreign key: Hosts(SubjectID) ⊆ Subject(SubjectID)
TaughtBy(SubjectID, ProfessorID)
foreign key: TaughtBy(SubjectID) ⊆ Subject(SubjectID)
foreign key: TaughtBy(ProfessorID) ⊆ Professor(ProfessorID)
Mentors(ProfessorID, StudentID)
foreign key: Mentors(ProfessorID) ⊆ Professor(ProfessorID)
foreign key: Mentors(StudentID) ⊆ Student(StudentID)

During the restructuring of the relational schema, I merged relations to facilitate access and eliminate unnecessary attributes. Specifically, I merged EmployedStudent into Student and UniversityProfessor and ExternalProfessor into Professor, simplifying the schema and reducing redundancy. I ensured the schema supports efficient access patterns by eliminating unnecessary joins and maintaining foreign key constraints to preserve referential integrity.