STUDENT ENROLLMENT MANAGEMENT SYSTEM

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, Date of the exam, ExamStartTime, and ExamEndTime. 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 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 phone numbers. For University Professors, we want to store the Department (e.g., Computer Science…). For External Professors, 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 must host at least one subject, and each subject is hosted in exactly one classroom. Each classroom can host multiple subjects, and we want to record which subjects are taught in each classroom.

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, ExamStartTime, ExamEndTime

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

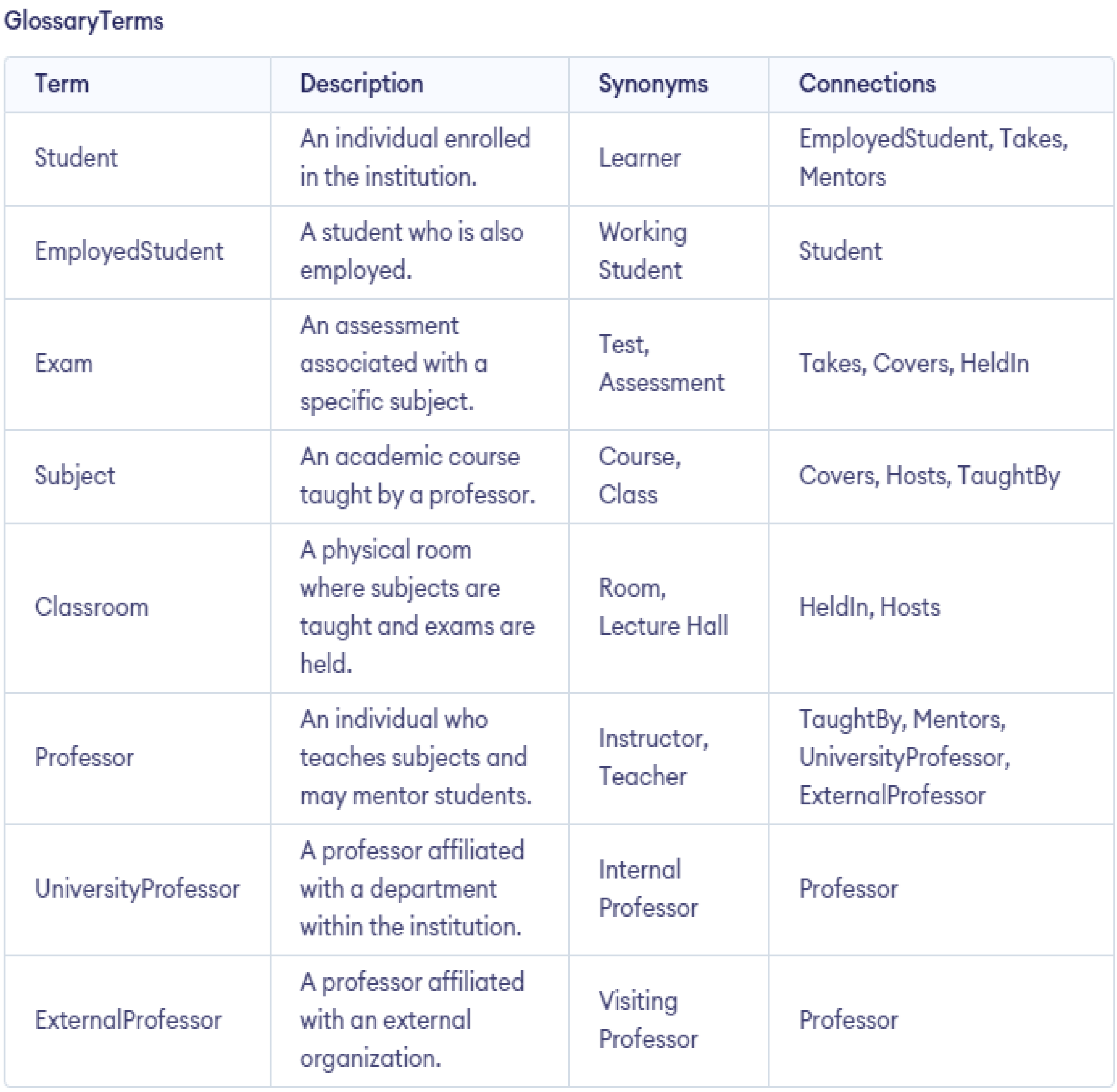
Hosts: Between Classroom and Subject

TaughtBy: Between Subject and Professor

Mentors: Between Professor and Student

b) Glossary of Terms

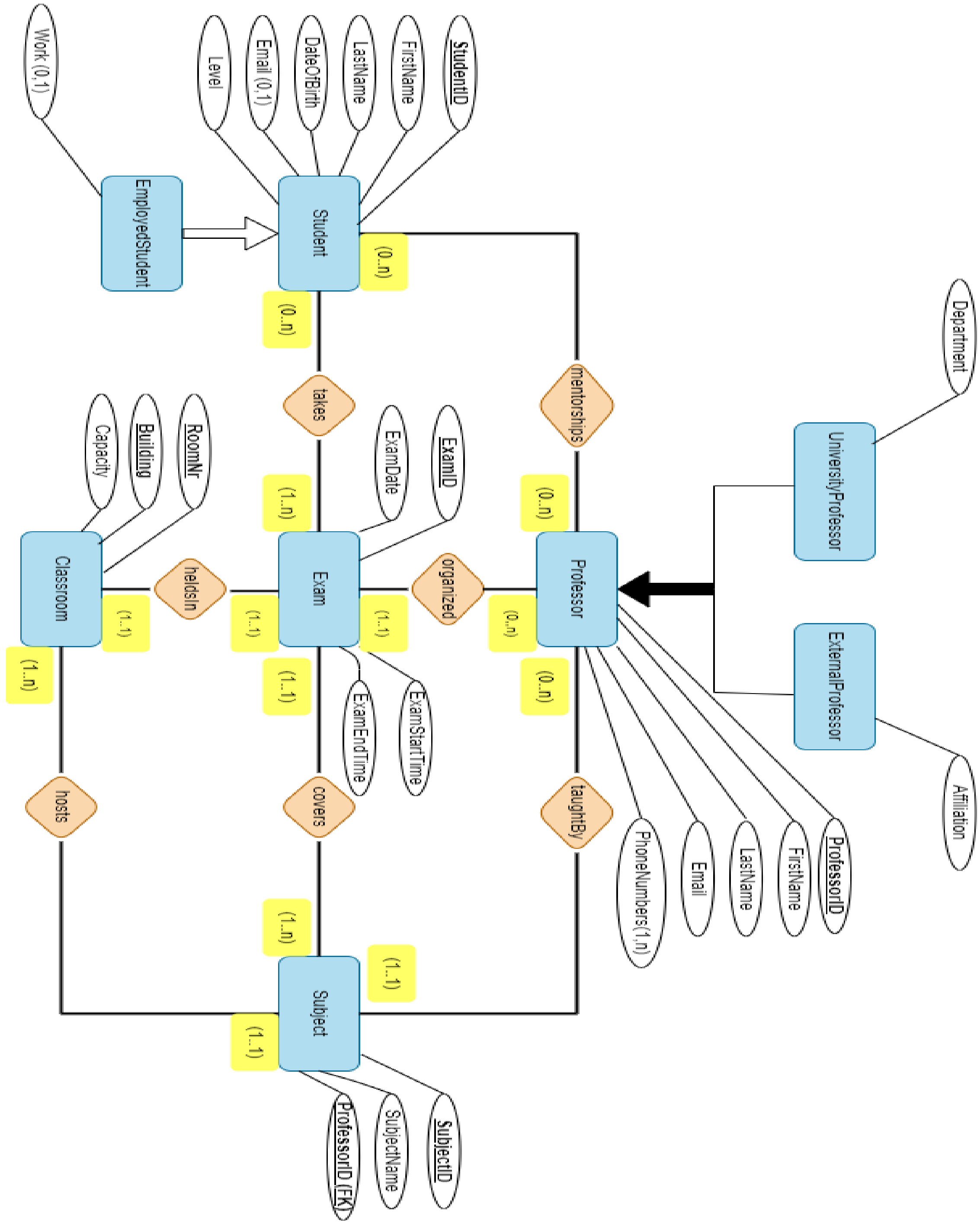
Glossary of terms provides definitions for specialized or technical words and phrases used within a particular field, subject, or document. It is designed to help readers understand the terminology and concepts that might be unfamiliar to them.



GlossaryTerms

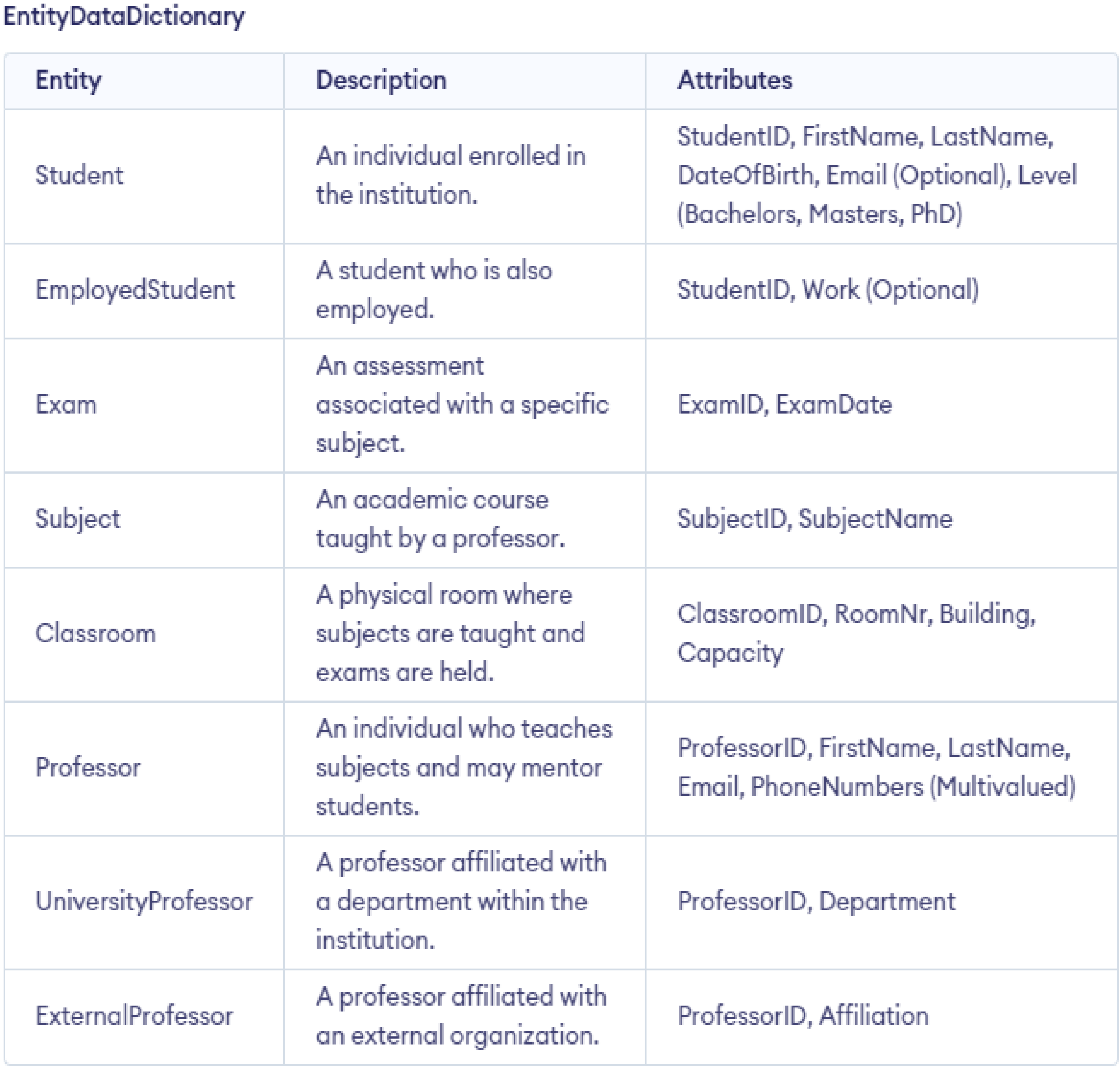
Term	Description	Synonyms	Connections
Student	An individual enrolled in the institution.	Learner	EmployedStudent, Takes, Mentors
EmployedStudent	A student who is also employed.	Working Student	Student
Exam	An assessment associated with a specific subject.	Test, Assessment	Takes, Covers, HeldIn
Subject	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



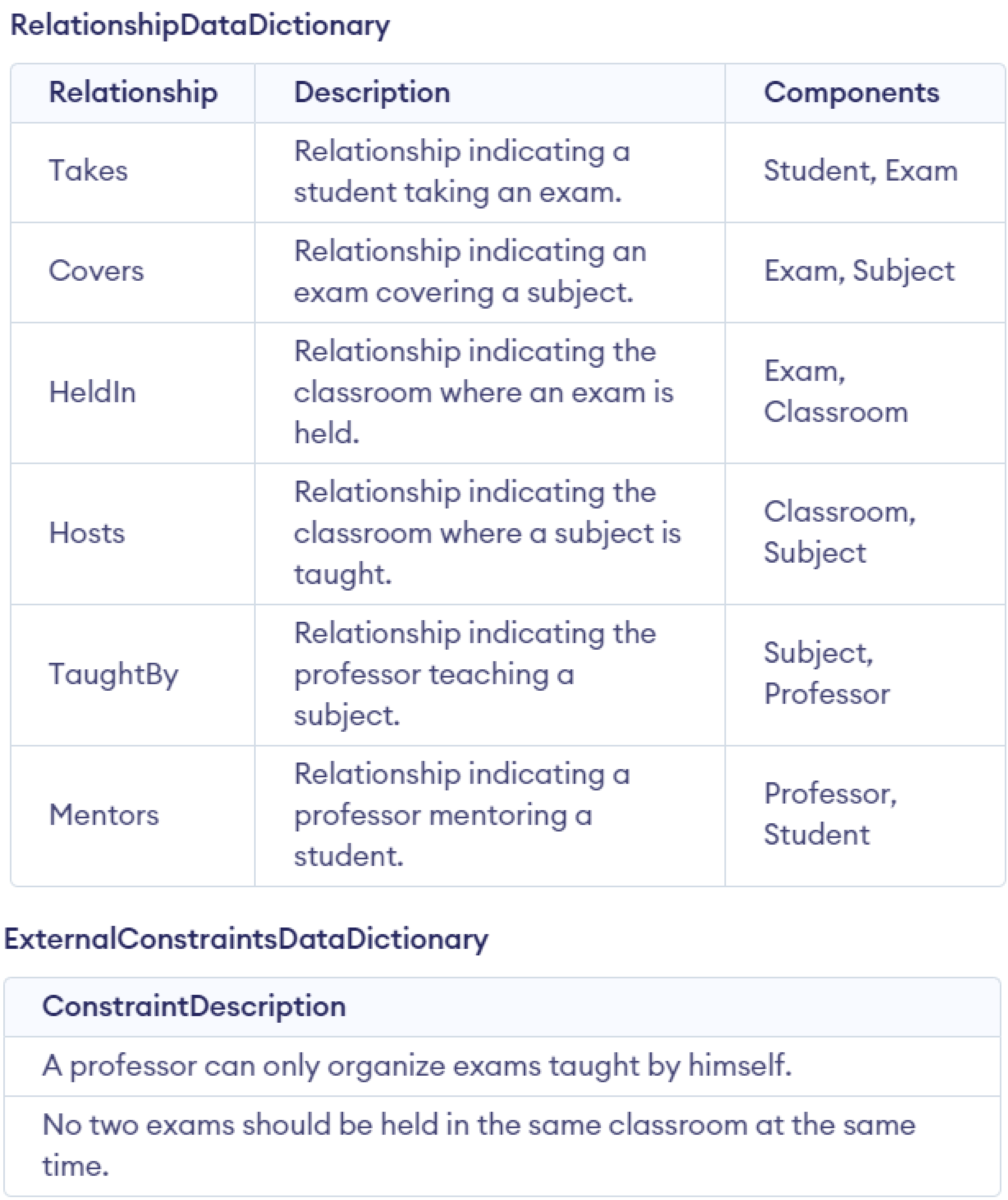
d) Data dictionary of the conceptual schema

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

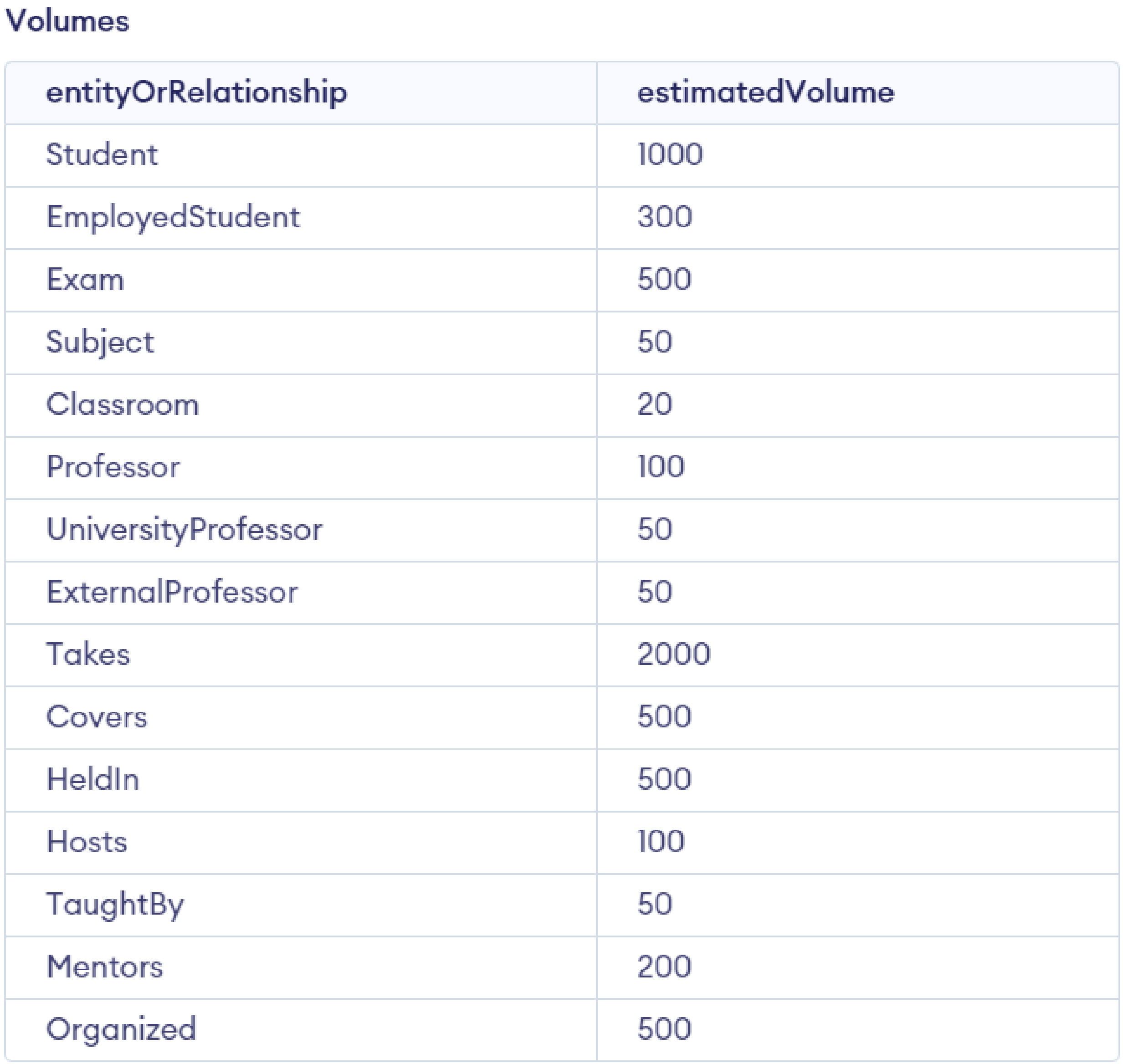
Relationship	Description	Components
Takes	Relationship indicating a student taking an exam.	Student, Exam
Covers	Relationship indicating an exam covering a subject.	Exam, Subject
HeldIn	Relationship indicating the classroom where an exam is held.	Exam, Classroom
Hosts	Relationship indicating the classroom 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

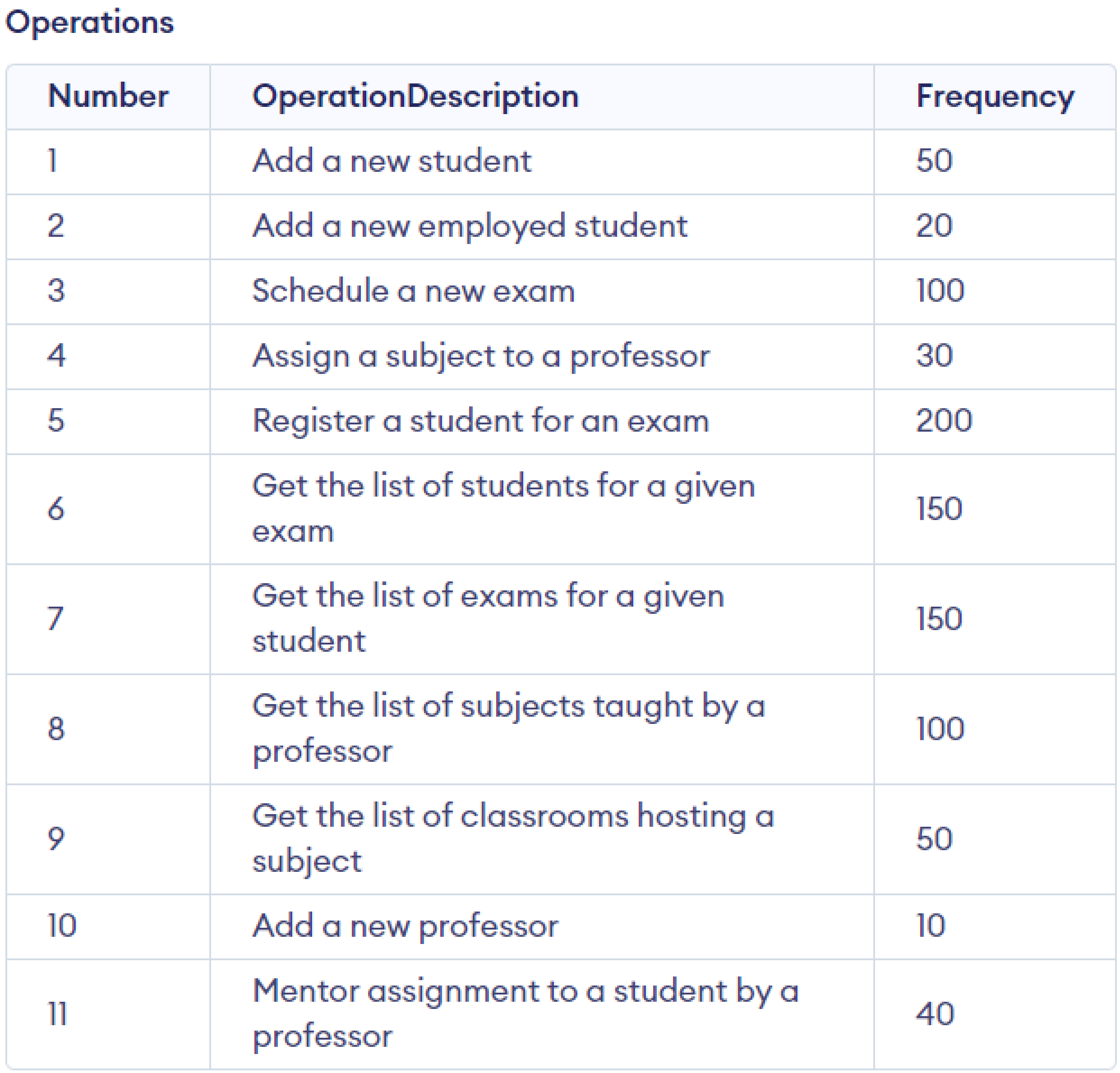
The Table of Volumes is a detailed estimate of the number of records for each entity and relationship in the database. It provides an approximation of how much data each table will hold, which helps in understanding the scale of the database and planning for storage, performance, and optimization needs.



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
Organized	500

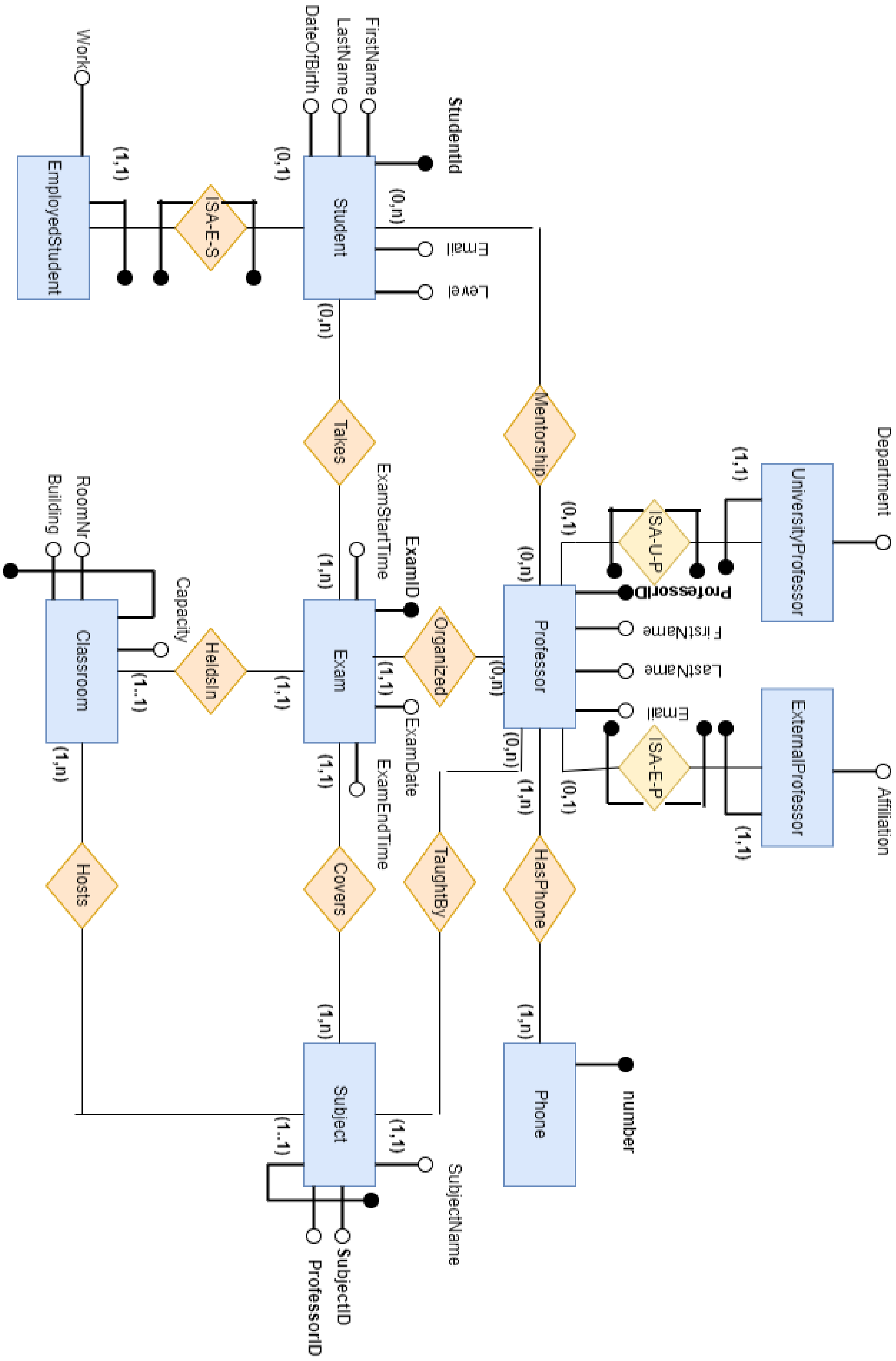
The Table of Operations lists the common database operations or queries that are expected to be performed, along with their expected frequency. This helps in understanding the workload on the database and is useful for optimizing performance and planning indexing strategies.



Operations

Number	OperationDescription	Frequency
1	Add a new student	50
2	Add a new employed student	20
3	Schedule a new exam	100
4	Assign a subject to a professor	30
5	Register a student for an exam	200
6	Get the list of students for a given exam	150
7	Get the list of exams for a given student	150
8	Get the list of subjects taught by a professor	100
9	Get the list of classrooms hosting a subject	50
10	Add a new professor	10
11	Mentor assignment to a student by a professor	40

2) Restructuring of the conceptual schema



a) Steps for Restructuring

1. **Initial Preparation:**

   I ensured that I had a complete picture of what is represented in the schema, so that no aspect of the conceptual schema was lost.

2. **Eliminating Multi-valued Attributes**:

   Introduced separate entities and relationships, such as creating the Phone entity for professor phone numbers, to handle multi-valued attributes.

   Ensured that all attributes were atomic.

3. **Elimination of ISA and Generalization**:
   Addressed ISA hierarchies and generalizations by adding constraints for attributes shared by child entities of the same parent entity and added the generalization constraints in the direct translation to the relational model.

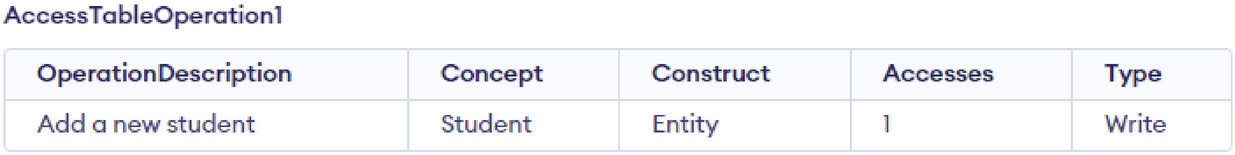
   Selected primary attributes or keys for each entity, such as ProfessorID for the Professor entity, to ensure each entity had a unique identifier.

4. **Reformulating Operations and Application Load Specifications**:

   Aligned operations and application load specifications with the restructured schema, detailing access types and frequencies in the access tables for each operation.

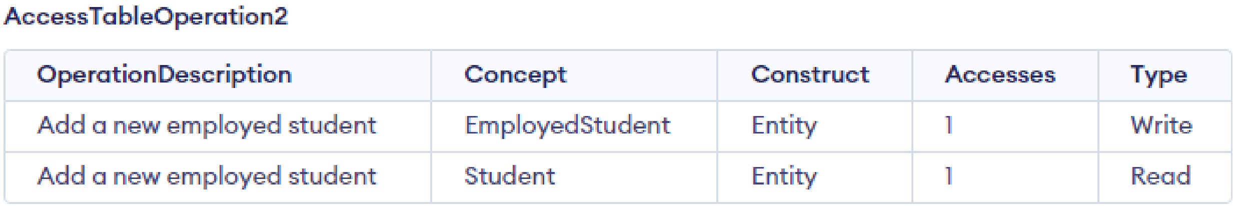
b) Cost of Evaluation

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.



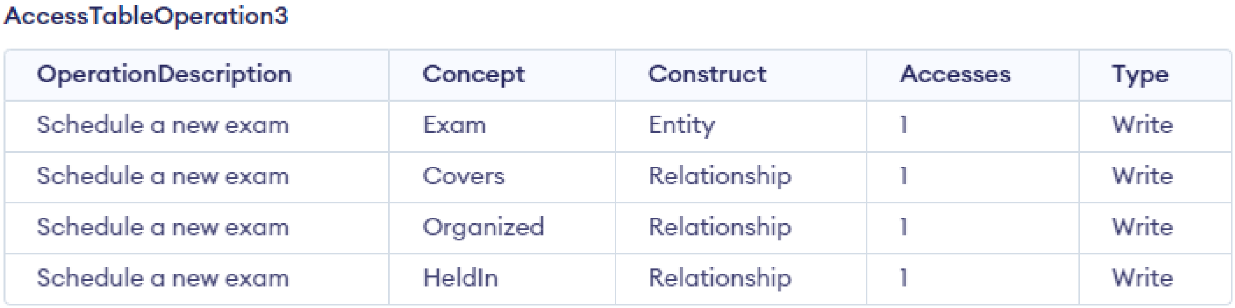
AccessTableOperation1

OperationDescription	Concept	Construct	Accesses	Type
Add a new student	Student	Entity	1	Write



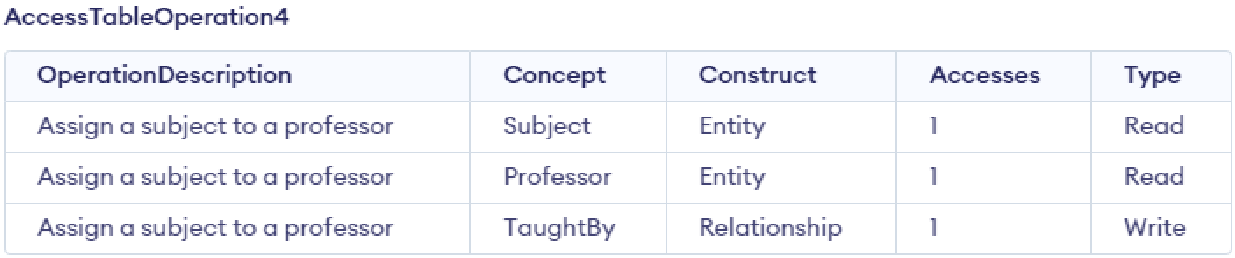
AccessTableOperation2

OperationDescription	Concept	Construct	Accesses	Type
Add a new employed student	EmployedStudent	Entity	1	Write
Add a new employed student	Student	Entity	1	Read



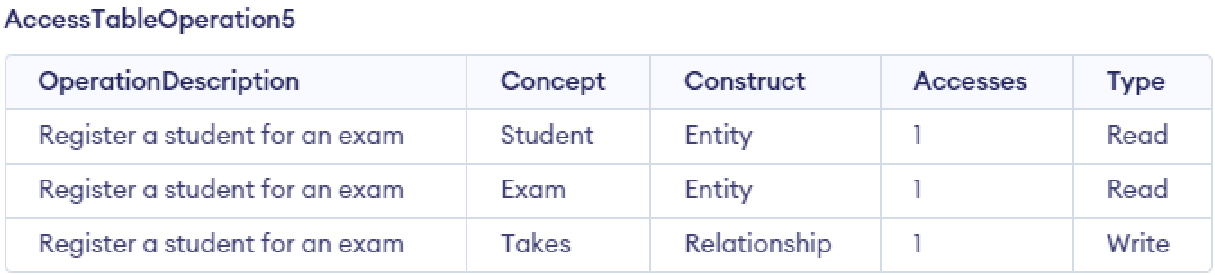
AccessTableOperation3

OperationDescription	Concept	Construct	Accesses	Type
Schedule a new exam	Exam	Entity	1	Write
Schedule a new exam	Covers	Relationship	1	Write
Schedule a new exam	Organized	Relationship	1	Write
Schedule a new exam	HeldIn	Relationship	1	Write



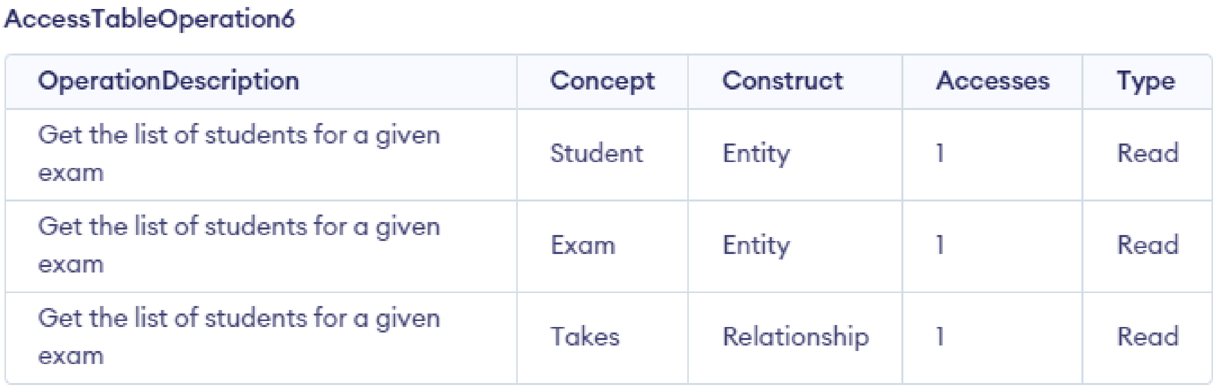
AccessTableOperation4

OperationDescription	Concept	Construct	Accesses	Type
Assign a subject to a professor	Subject	Entity	1	Read
Assign a subject to a professor	Professor	Entity	1	Read
Assign a subject to a professor	TaughtBy	Relationship	1	Write



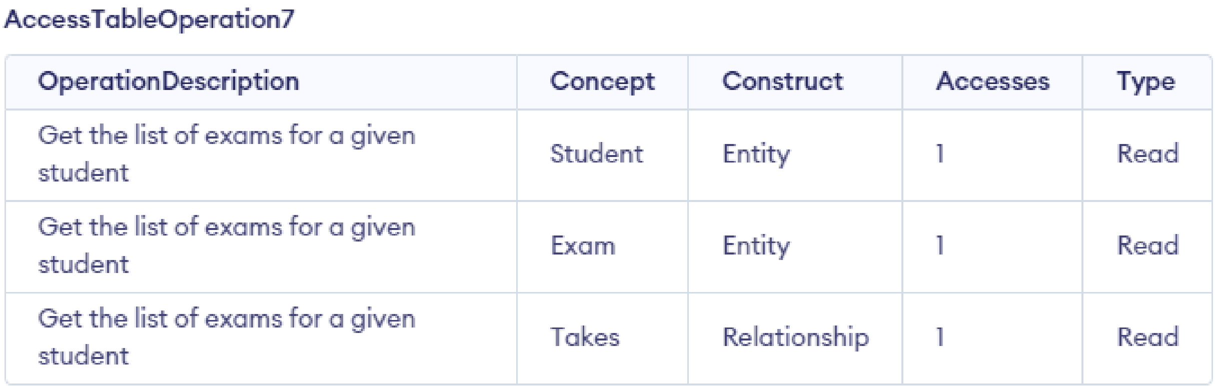
AccessTableOperation5

OperationDescription	Concept	Construct	Accesses	Type
Register a student for an exam	Student	Entity	1	Read
Register a student for an exam	Exam	Entity	1	Read
Register a student for an exam	Takes	Relationship	1	Write



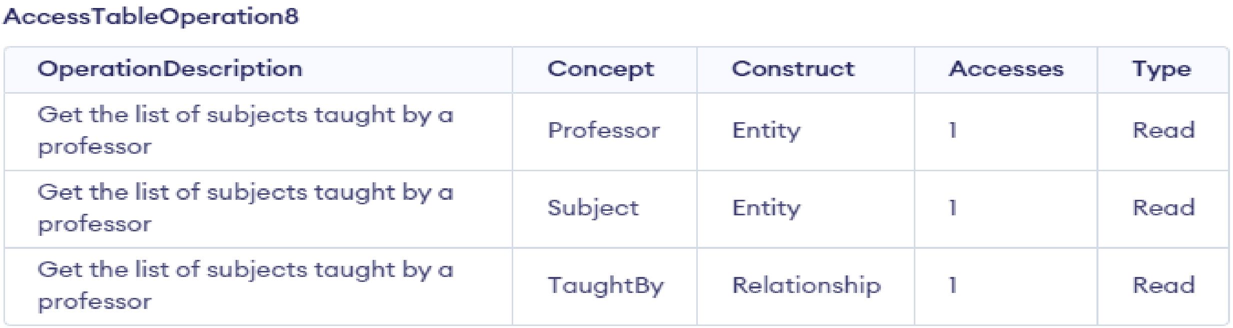
AccessTableOperation6

OperationDescription	Concept	Construct	Accesses	Type
Get the list of students for a given exam	Student	Entity	1	Read
Get the list of students for a given exam	Exam	Entity	1	Read
Get the list of students for a given exam	Takes	Relationship	1	Read



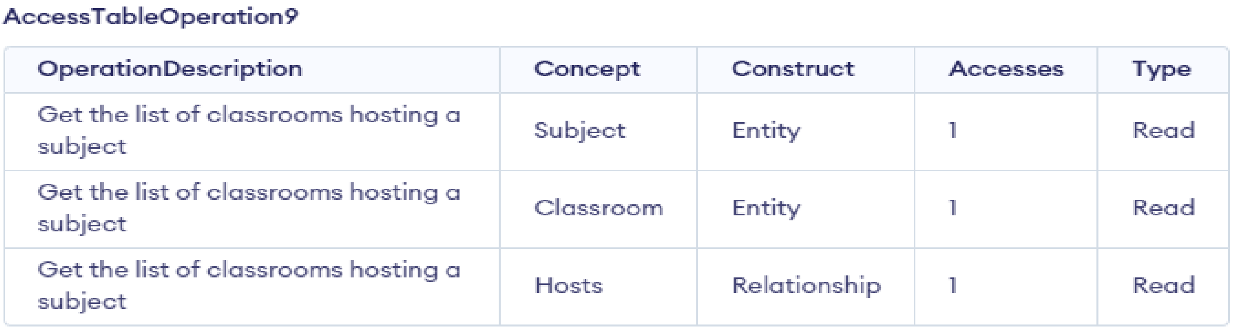
AccessTableOperation7

OperationDescription	Concept	Construct	Accesses	Type
Get the list of exams for a given student	Student	Entity	1	Read
Get the list of exams for a given student	Exam	Entity	1	Read
Get the list of exams for a given student	Takes	Relationship	1	Read



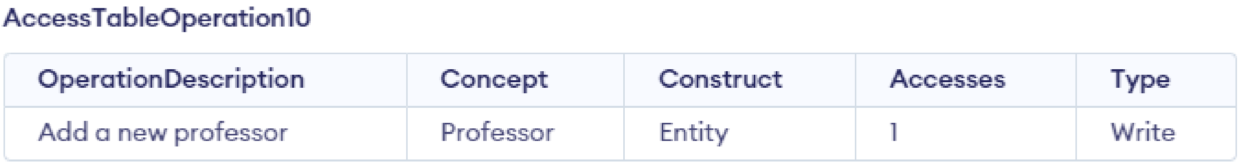
AccessTableOperation8

OperationDescription	Concept	Construct	Accesses	Type
Get the list of subjects taught by a professor	Professor	Entity	1	Read
Get the list of subjects taught by a professor	Subject	Entity	1	Read
Get the list of subjects taught by a professor	TaughtBy	Relationship	1	Read



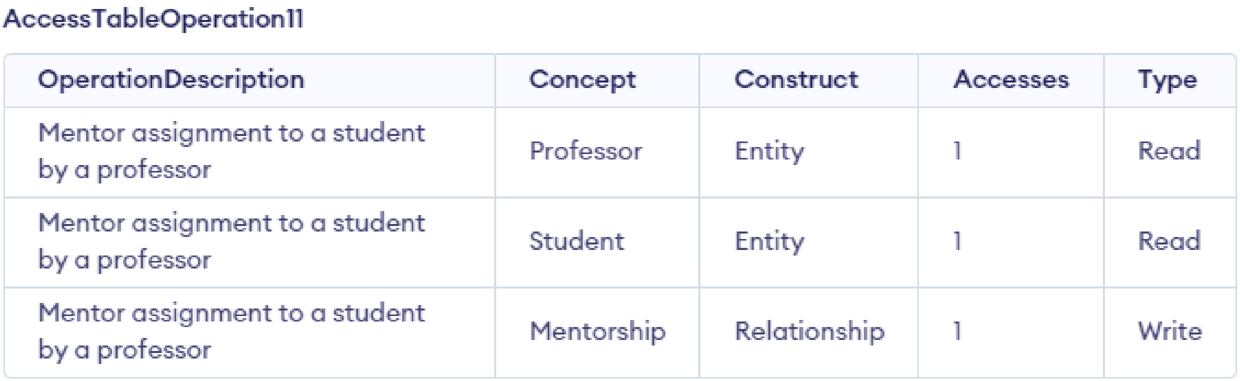
AccessTableOperation9

OperationDescription	Concept	Construct	Accesses	Type
Get the list of classrooms hosting a subject	Subject	Entity	1	Read
Get the list of classrooms hosting a subject	Classroom	Entity	1	Read
Get the list of classrooms hosting a subject	Hosts	Relationship	1	Read



AccessTableOperation10

OperationDescription	Concept	Construct	Accesses	Type
Add a new professor	Professor	Entity	1	Write



AccessTableOperation11

OperationDescription	Concept	Construct	Accesses	Type
Mentor assignment to a student by a professor	Professor	Entity	1	Read
Mentor assignment to a student by a professor	Student	Entity	1	Read
Mentor assignment to a student by a professor	Mentorship	Relationship	1	Write

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, ExamStartTime, ExamEndTime)
Organized(**ExamID**, ProfessorID)
foreign key: Organized(ExamID) ⊆ Exam(ExamID)
foreign key: Organized(ProfessorID) ⊆ Professor(ProfessorID)
Subject(**SubjectID**, **ProfessorID,** SubjectName,)
foreign key: Subject(ProfessorID) ⊆ Professor(ProfessorID)
TaughtBy(**SubjectID**, **ProfessorID**)
foreign key: TaughtBy(SubjectID) ⊆ Subject(SubjectID)
foreign key: TaughtBy(ProfessorID) ⊆ Professor(ProfessorID)
Classroom(**RoomNr**, **Building**, Capacity)
Takes(**StudentID**, **ExamID**)
foreign key: Takes(StudentID) ⊆ Student(StudentID)
foreign key: Takes(ExamID) ⊆ Exam(ExamID)
Inclusion: Exam(ExamID) ⊆ Takes(ExamID)
Covers(**ExamID**, SubjectID)
foreign key: Covers(ExamID) ⊆ Exam(ExamID)
foreign key: Covers(SubjectID) ⊆ Subject(SubjectID)
HeldIn(**ExamID**, RoomNr, Building)
foreign key: HeldIn(ExamID) ⊆ Exam(ExamID)
foreign key: HeldIn(RoomNr, Building) ⊆ Classroom(RoomNr, Building)
Hosts(**RoomNr**, **Building**, **SubjectID**)
foreign key: Hosts(RoomNr, Building) ⊆ Classroom(RoomNr, Building)
foreign key: Hosts(SubjectID) ⊆ Subject(SubjectID)
Inclusion: Classroom(RoomNr, Building) ⊆ Hosts(RoomNr, Building)
Inclusion: Subject(SubjectID) ⊆ Hosts(SubjectID)
Mentors(**ProfessorID**, **StudentID**)
foreign key: Mentors(ProfessorID) ⊆ Professor(ProfessorID)
foreign key: Mentors(StudentID) ⊆ Student(StudentID)

Generalization constraint:
UniversityProfessor[ProfessorID]∩ExternalProfessor[ProfessorID]=∅
Professor[ProfessorID]⊆UniversityProfessor[ProfessorID]∪ExternalProfessor[ProfessorID]

4) Restructuring of the relational schema

-- Student Table with optional attribute Work

Student(**StudentID**, FirstName, LastName, DateOfBirth, Email, Level, *Work)

-- Professor Table with optional attributes Department and Affiliation

Professor(**ProfessorID**, FirstName, LastName, Email, *Department, *Affiliation)

-- Phone Table

Phone(**Number**, **ProfessorID**)

foreign key: Phone(ProfessorID) ⊆ Professor(ProfessorID)

-- Exam Table

Exam(**ExamID**, ExamDate, ExamStartTime, ExamEndTime, ProfessorID, SubjectID, RoomNr, Building)

foreign key: Exam(ProfessorID) ⊆ Professor(ProfessorID)

foreign key: Exam(SubjectID) ⊆ Subject(SubjectID)

foreign key: Exam(RoomNr, Building) ⊆ Classroom(RoomNr, Building)

-- Subject Table

Subject(**SubjectID**, **ProfessorID**,SubjectName)

foreign key: Subject(ProfessorID) ⊆ Professor(ProfessorID)

-- TaughtBy Table

TaughtBy(**SubjectID**, **ProfessorID**)

foreign key: TaughtBy(SubjectID) ⊆ Subject(SubjectID)

foreign key: TaughtBy(ProfessorID) ⊆ Professor(ProfessorID)

-- Classroom Table

Classroom(**RoomNr**, **Building**, Capacity)

-- Takes Table

Takes(**StudentID**, **ExamID**)

foreign key: Takes(StudentID) ⊆ Student(StudentID)

foreign key: Takes(ExamID) ⊆ Exam(ExamID)

-- Hosts Table

Hosts(**RoomNr**, **Building**, **SubjectID**)

foreign key: Hosts(RoomNr, Building) ⊆ Classroom(RoomNr, Building)

foreign key: Hosts(SubjectID) ⊆ Subject(SubjectID)

-- Mentors Table

Mentors(**ProfessorID**, **StudentID**)

foreign key: Mentors(ProfessorID) ⊆ Professor(ProfessorID)

foreign key: Mentors(StudentID) ⊆ Student(StudentID)

-- Inclusion Constraints

Inclusion: Exam(ExamID) ⊆ Takes(ExamID)

Inclusion: Classroom(RoomNr, Building) ⊆ Hosts(RoomNr, Building)

Inclusion: Subject(SubjectID) ⊆ Hosts(SubjectID)