Relational Calculus

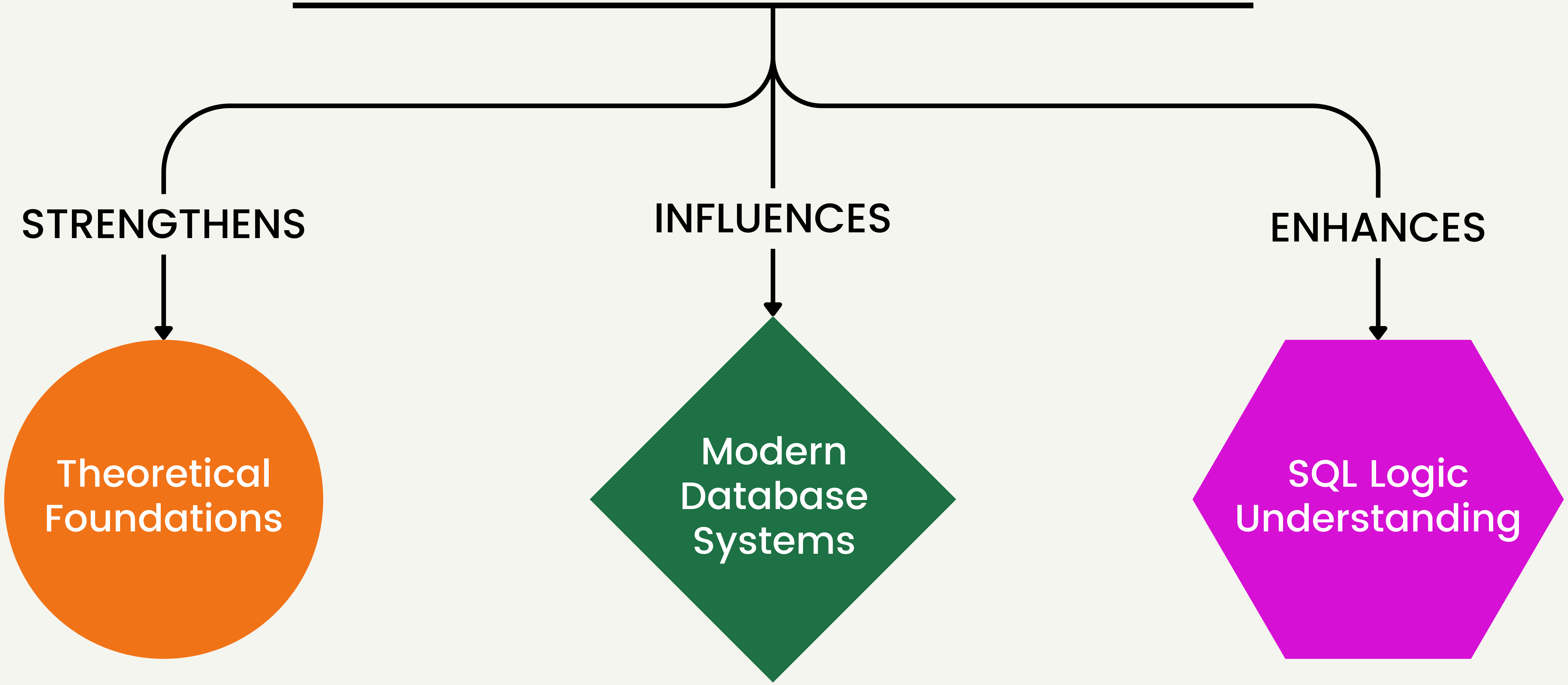
in Database Management Systems

Presented by
Om Gupta, Vikash Kumar, Aditya Raj,
Parkhi Jain, Adarsh Sharma & Gurnam Singh

BOOK REFERRED:

Elmasri, R. (2008).
Fundamentals of Database Systems.
Pearson Education India.
6th Edition

Why we need Relational Calculus?



STRENGTHENS

Theoretical Foundations

INFLUENCES

Modern Database Systems

ENHANCES

SQL Logic Understanding

What is Relational Calculus?

It is a **Non-Procedural Query Language.**

We specify **what** we want rather than **how** to retrieve it. Hence, also called **declarative language**.

We indirectly use it to query a database, that is, **retrieve** data from a database.

What is Relational Calculus?

It is the "**Grammar**" of databases.

Just like natural languages have different ways to form sentences, databases have two forms of Relational Calculus:

- **Tuple Relational Calculus (TRC):** Describes what tuples (rows) satisfy a given condition.
- **Domain Relational Calculus (DRC):** Works with domain values rather than entire tuples.

What is Relational Calculus?

It is based on **Predicate Calculus**.

- Relational calculus uses **predicates** and **quantifiers** to express conditions on data.
- Queries are written as logical statements, specifying what data to retrieve rather than how to retrieve it.

Example query:

$$\{\underbrace{t}_{\substack{\textbf{Result}\\ \text{(Tuple variable `t')}}} \mid \underbrace{t \in \text{Student} \land t.\text{Age} > 20}_{\substack{\textbf{Predicate}\\ \text{(Condition used to fetch tuple `t')}}}\}$$

Relational...

Calculus V/S Algebra

Non-Procedural Query Language	**Procedural** Query Language
Specifies ***what*** data to retrieve (without specifying steps)	Specifies ***how*** to retrieve data (step-by-step operations)
Uses **logical predicates** in order to define conditions	Uses **operations** like Selection, Projection, Join, Union, Difference, etc
Query defined by **logical formulas** (predicate calculus)	Query defined by a **sequence of operations** on relations
More **abstract**, closer to first-order logic	More **intuitive**, closer to set operations.
SQL query **formulation** (internally) is closer to relational calculus	SQL **execution** (internally) is closer to relational algebra

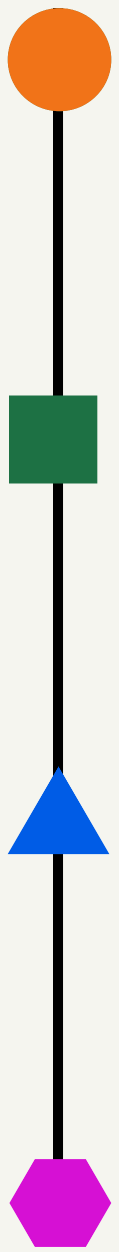
FIND NAMES OF ALL STUDENTS ENROLLED IN COURSE C1

$\{\, s.\text{name} \mid s \in \text{Student} \wedge \exists e\, (e \in \text{Enrolled} \wedge s.\text{sid} = e.\text{sid} \wedge e.\text{cid} =' C1') \,\}$

$\pi_{\text{name}}\left(\sigma_{\text{Enrolled.cid}='C1'}(\text{Student} \bowtie \text{Enrolled})\right)$

Basics of **Predicate Calculus**

used in the following presentation



COMPARISON OPERATORS

$<, \leq, =, \neq, >, \geq$

CONNECTIVES

and ($\wedge$), or ($\vee$), not ($\neg$)

IMPLICATION

$P1 \Rightarrow P2$

QUANTIFIERS

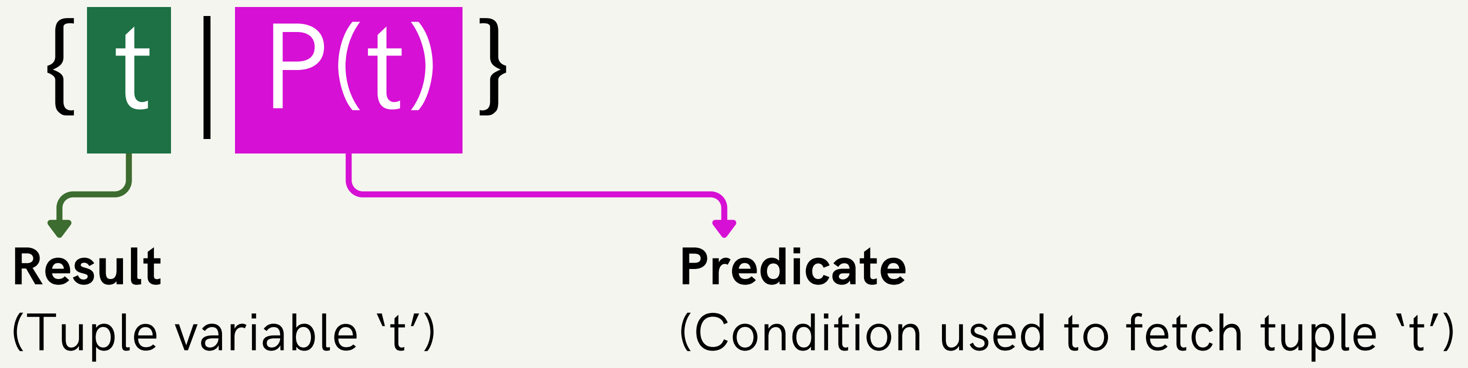
Universal: $\forall\ t \in r\ (Q(t))$

Existential: $\exists\ t \in r\ (Q(t))$

Tuple Relational Calculus (TRC)

Selecting **tuples** in a relation that satisfy a given condition (or predicate).

- The **result** of the query can consist of one or more tuples
- Basic **syntax**:



$\{ t \mid P(t) \}$

Result
(Tuple variable ‘t’)

Predicate
(Condition used to fetch tuple ‘t’)

- **Result:** Set of tuples ‘**t**’ such that predicate ‘**P**’ is true for ‘**t**’.
- **Notations:**
  - `t: Tuple Variable`
  - `t.A: Value of t on attribute A`
  - `t ∈ r: Denotes that tuple t is in relation r`
  - `P: predicate i.e, the condition on tuples`

NOTE

Variables with quantifiers are **bound** variables, all others are free.

$\{ t \mid P(t) \}$

must be the **only free variable.**

DATABASE

used in the following presentation

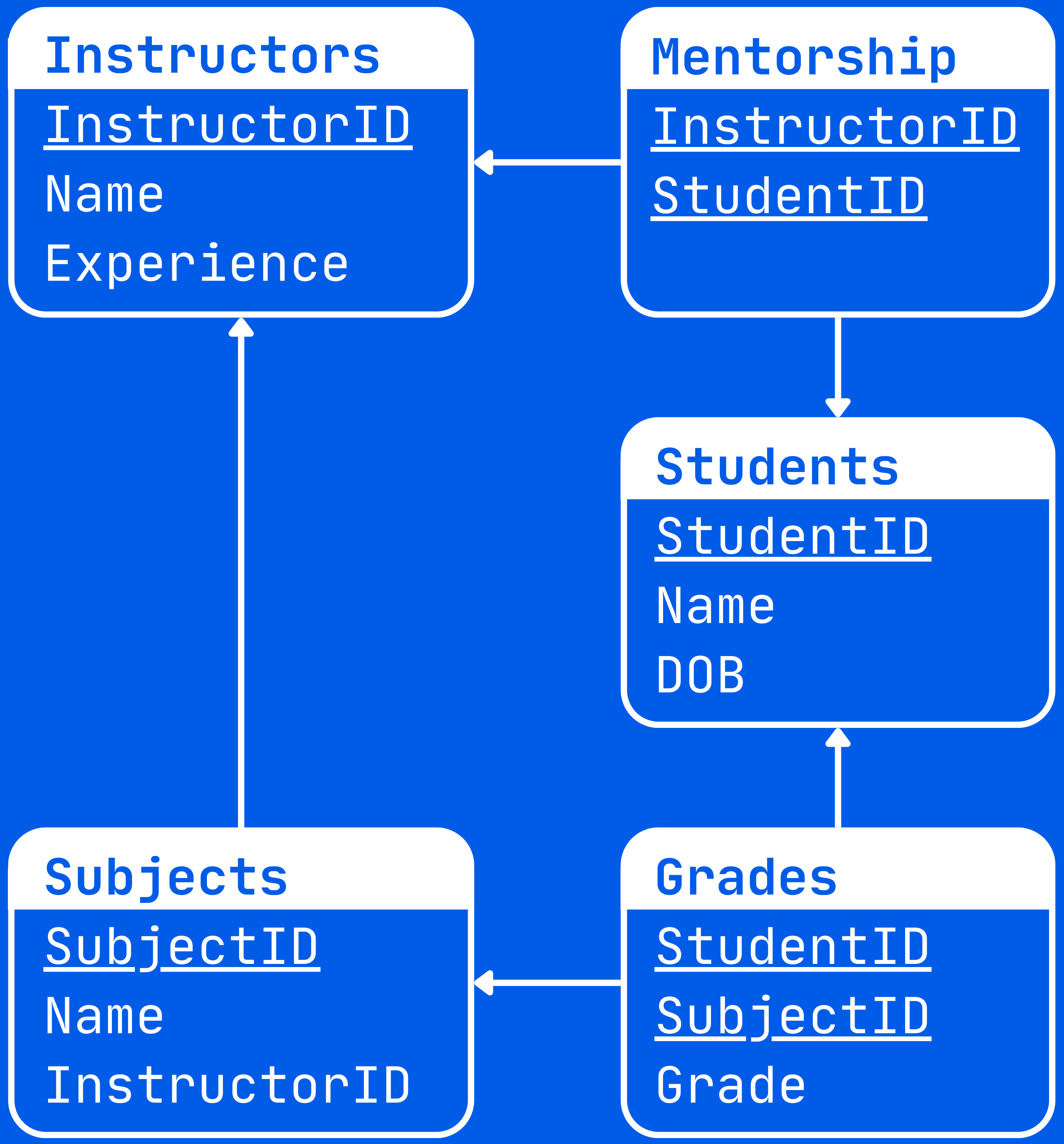
Students: (StudentID, Name, DOB)

Instructors: (InstructorID, Name, Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

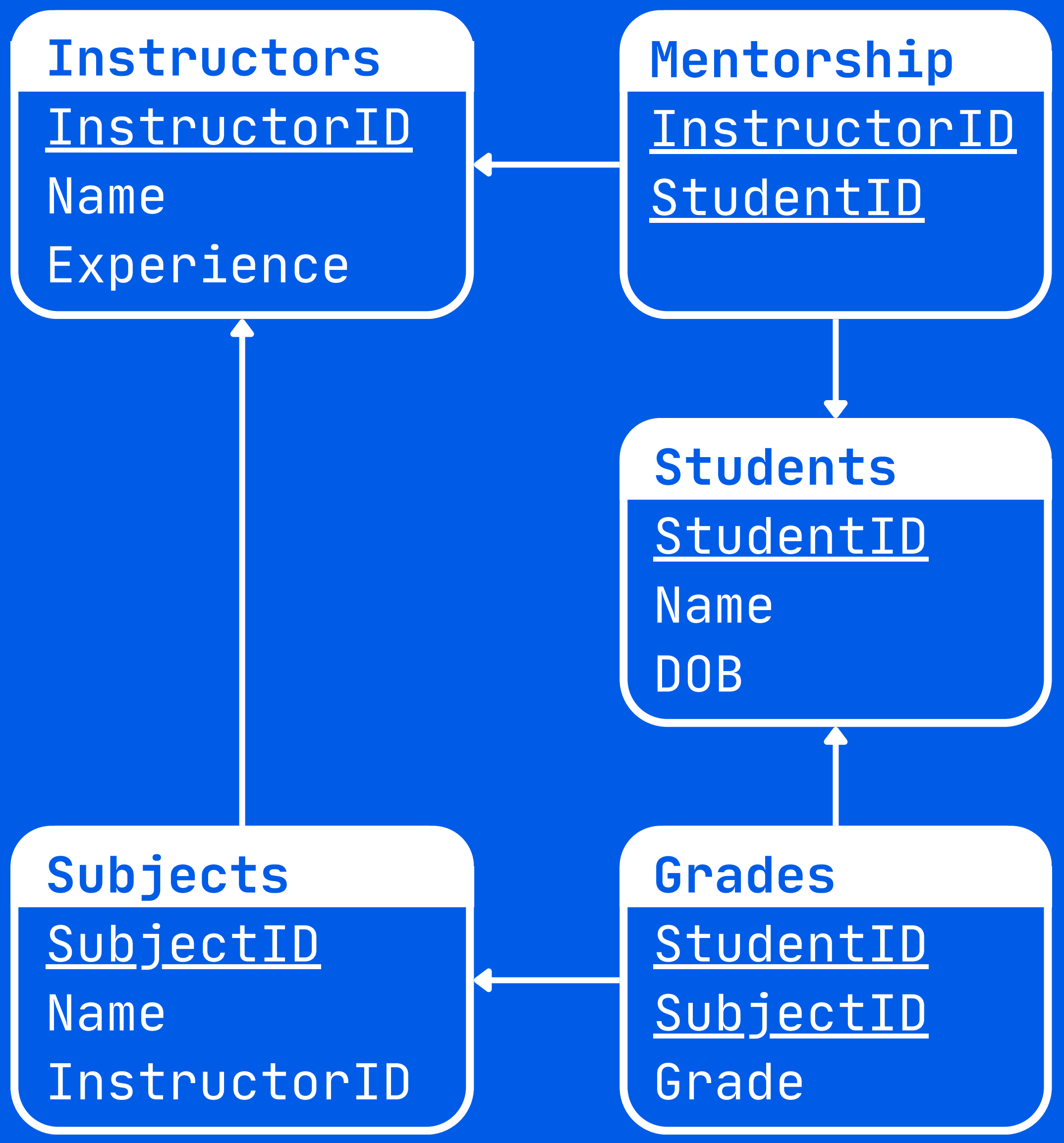
Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

TRC Queries

1 Retrieve details of all the students from the Students table.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

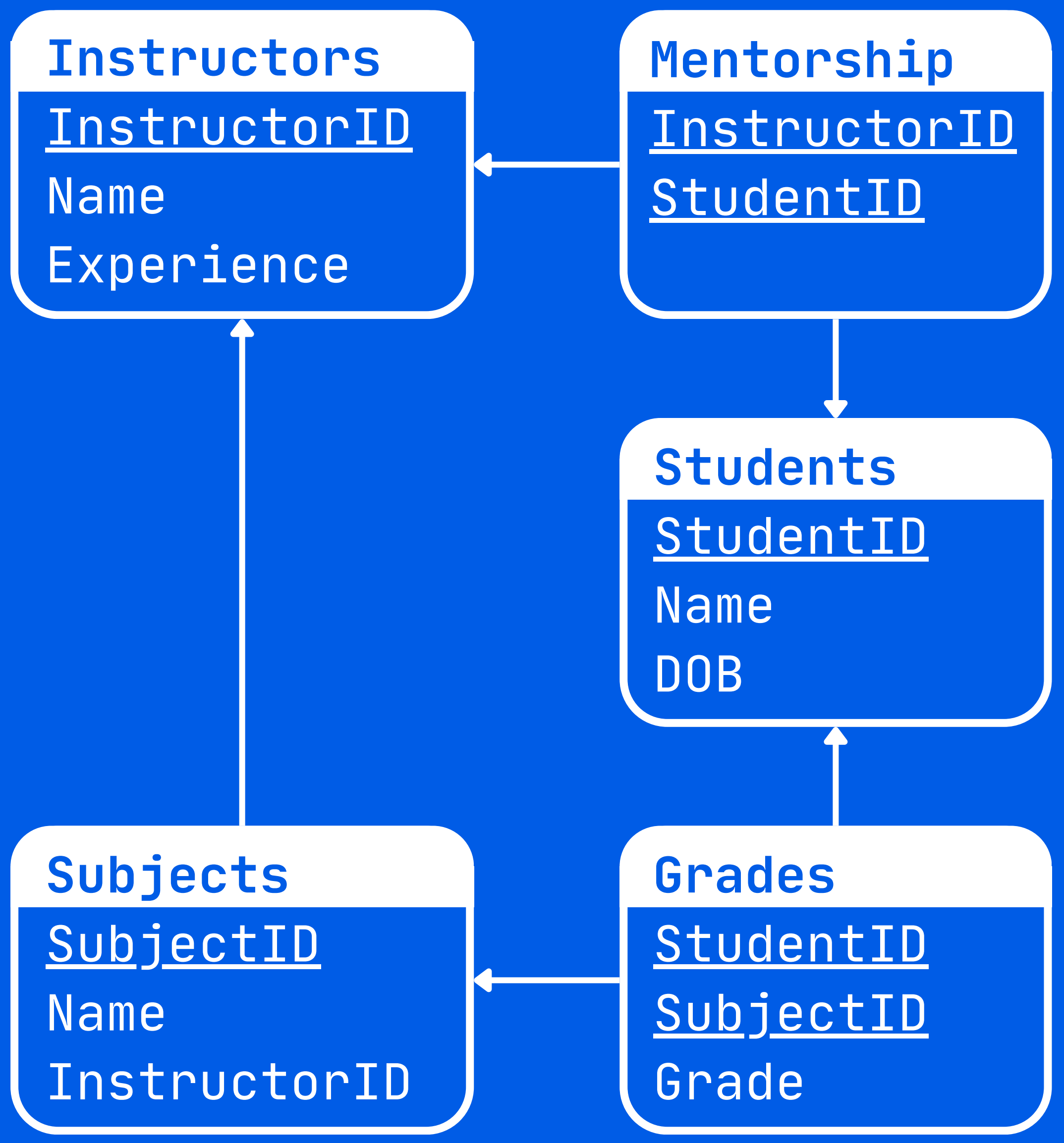
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

1 Retrieve details of all the students from the Students table.

$$\{t \mid t \in \textit{Students}\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

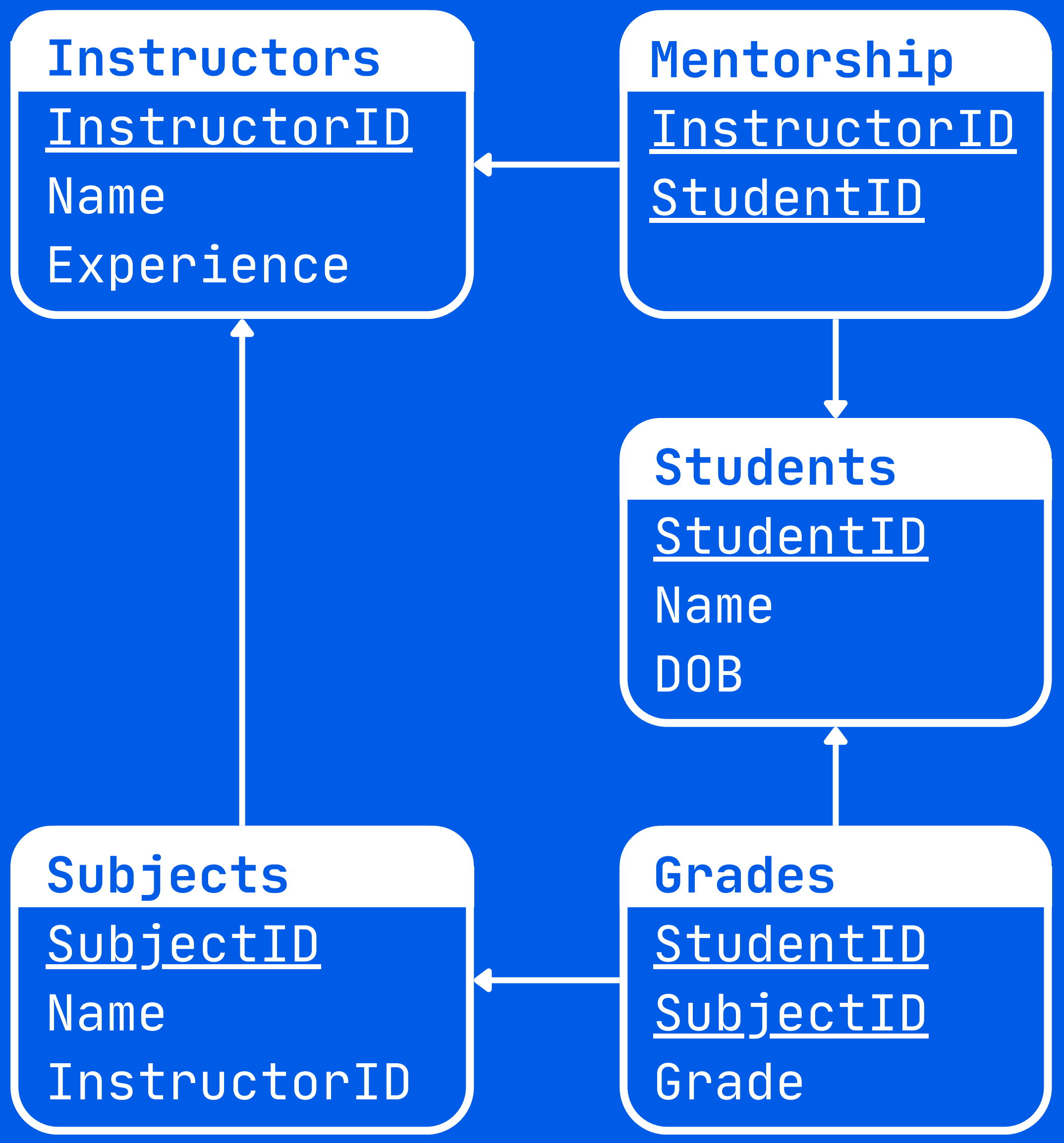
Mentorship: (InstructorID, StudentID)

TRC Queries

1 Retrieve details of all the students from the Students table.

$$\{t \mid t \in \mathit{Students}\}$$

2 Retrieve the names and DOB of all the students.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

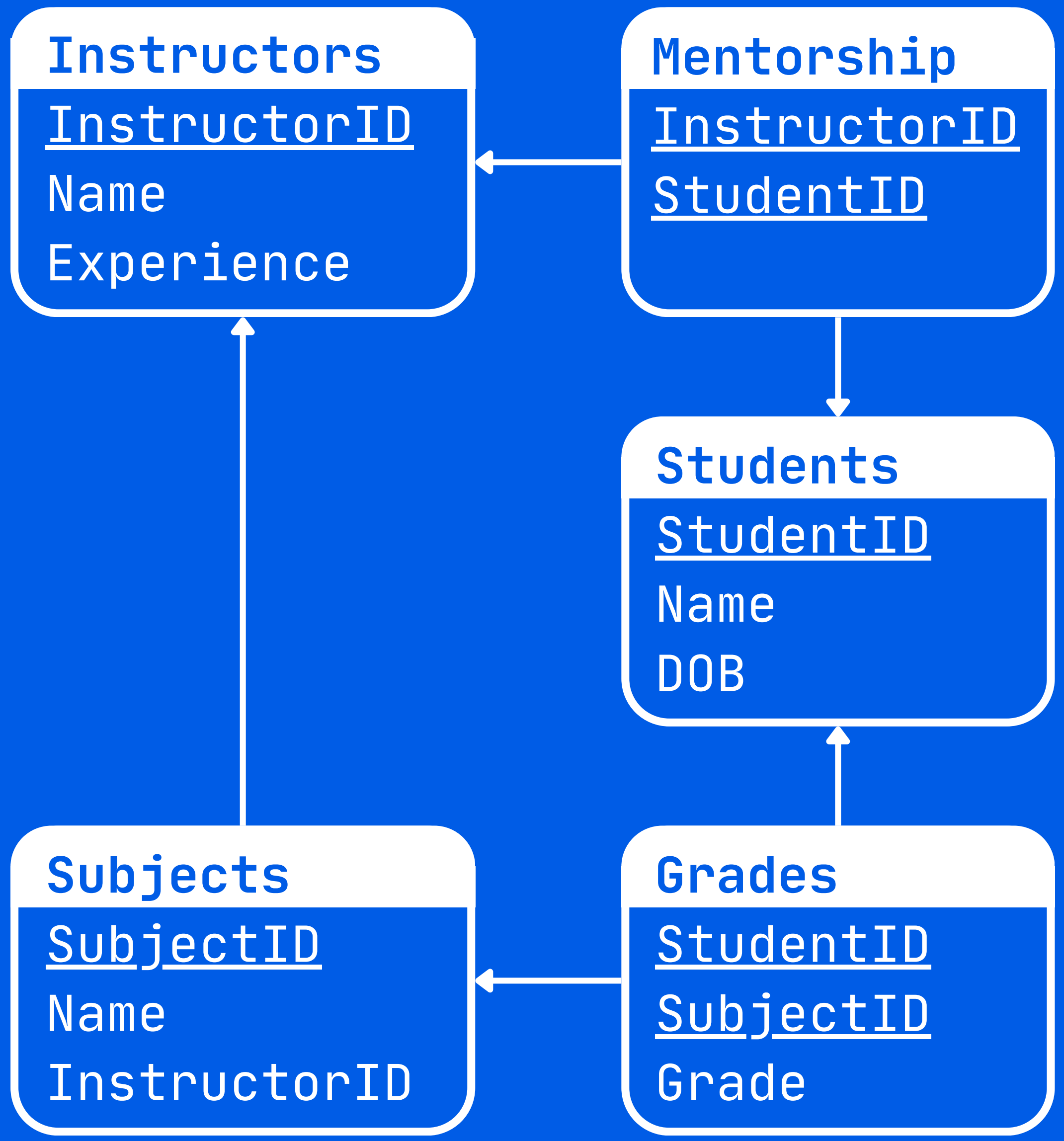
TRC Queries

1 Retrieve details of all the students from the Students table.

$$\{t \mid t \in \mathit{Students}\}$$

2 Retrieve the names and DOB of all the students.

$$\{t.\text{Name}, t.\text{DOB} \mid t \in \mathit{Students}\}$$



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

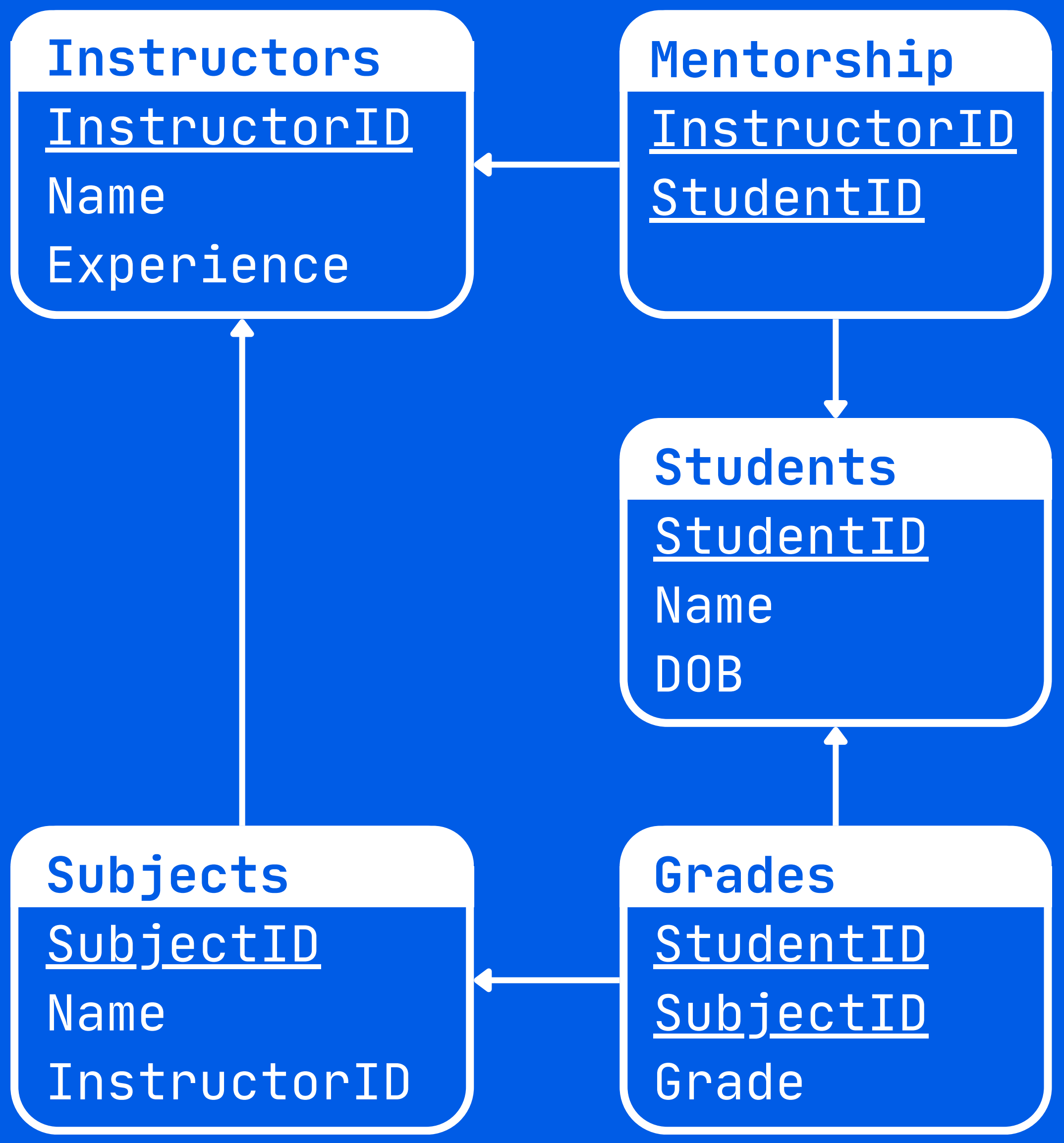
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

3 Find StudentID of students who scored grade “A”.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

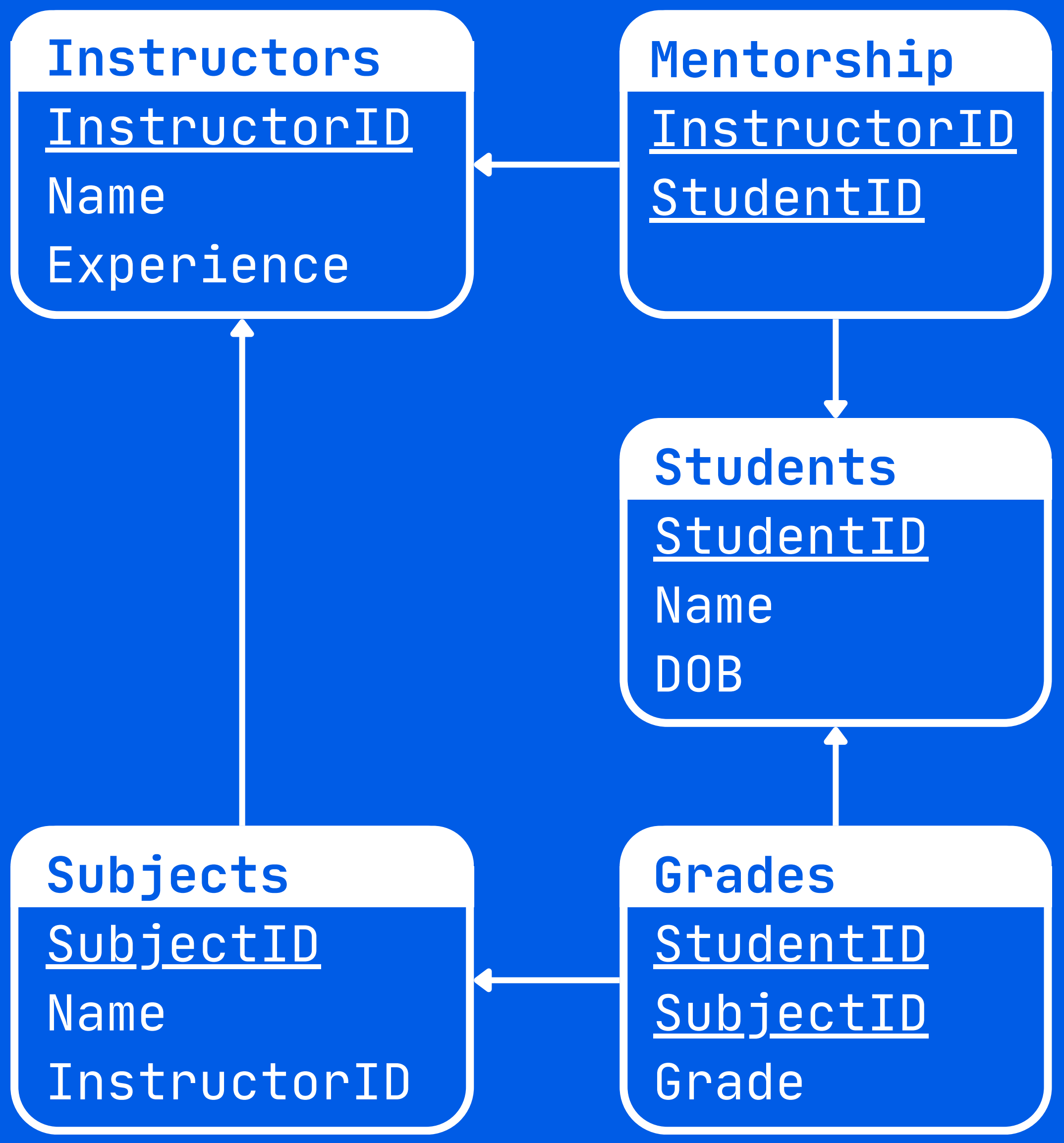
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

3 Find StudentID of students who scored grade “A”.

$$\{t.StudentID \mid t \in Grades \land t.Grade = “A” \}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

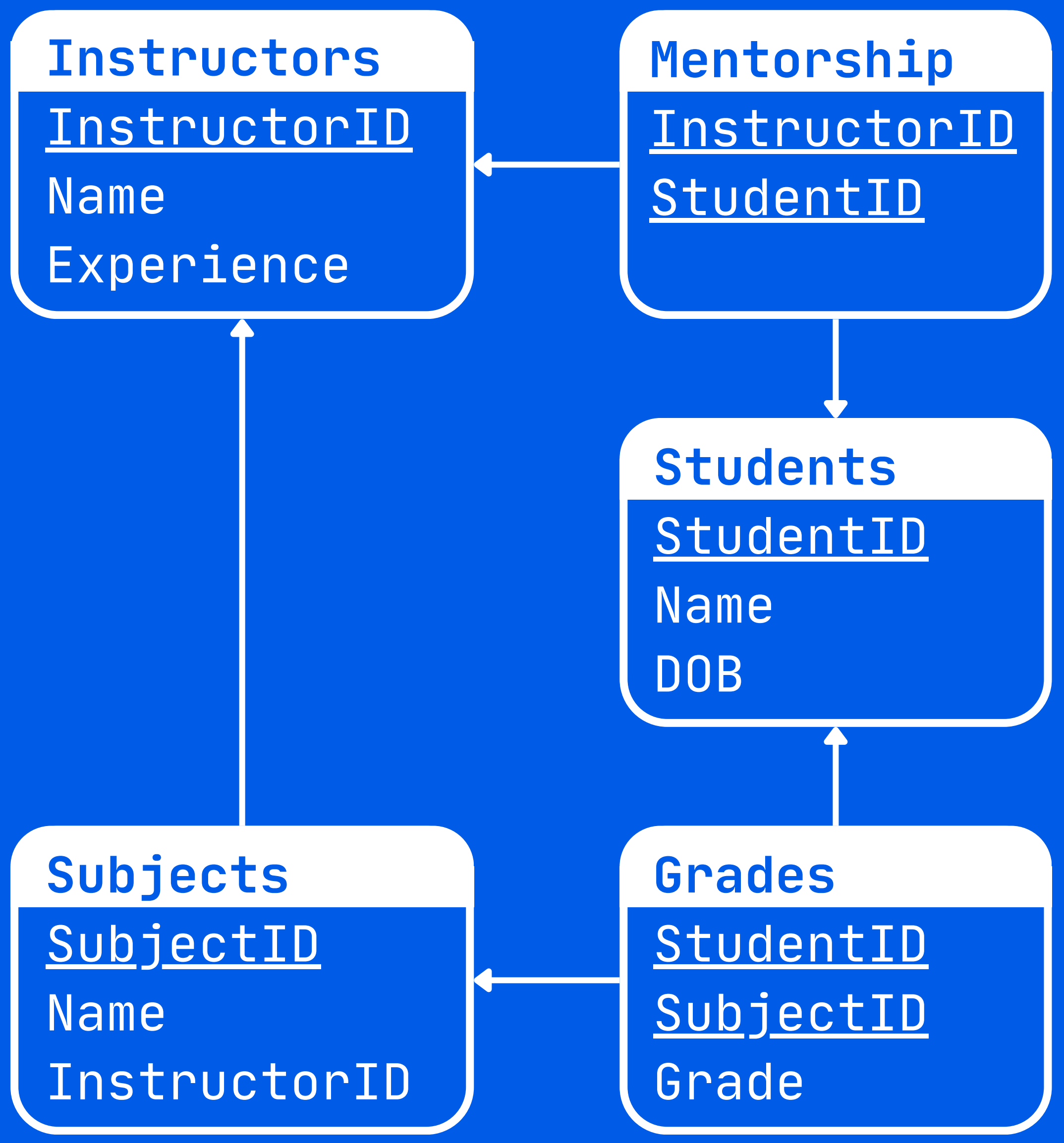
Mentorship: (InstructorID, StudentID)

TRC Queries

3 Find StudentID of students who scored grade “A”.

$$\{t.StudentID \mid t \in Grades \land t.Grade = \text{“A”}\}$$

4 Retrieve the instructor ID and the Name from the Instructor table who have more than 5 year of experience.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

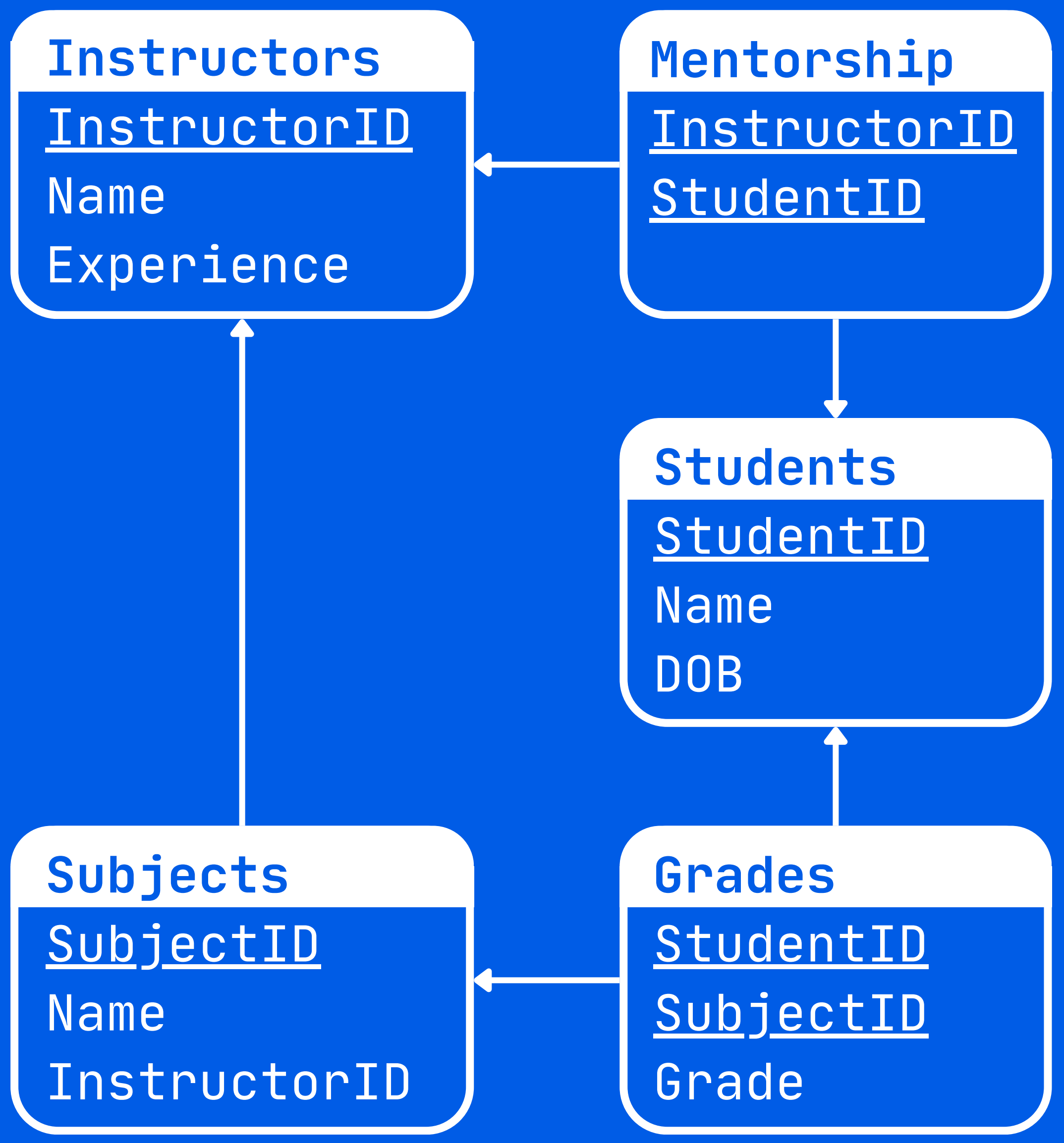
TRC Queries

3 Find StudentID of students who scored grade “A”.

$$\{t.\text{StudentID} \mid t \in \text{Grades} \wedge t.\text{Grade} = \text{“A”} \}$$

4 Retrieve the instructor ID and the Name from the Instructor table who have more than 5 year of experience.

$$\{t.\text{InstructorID}, t.\text{Name} \mid t \in \text{Instructors} \wedge t.\text{Experience} > 5\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

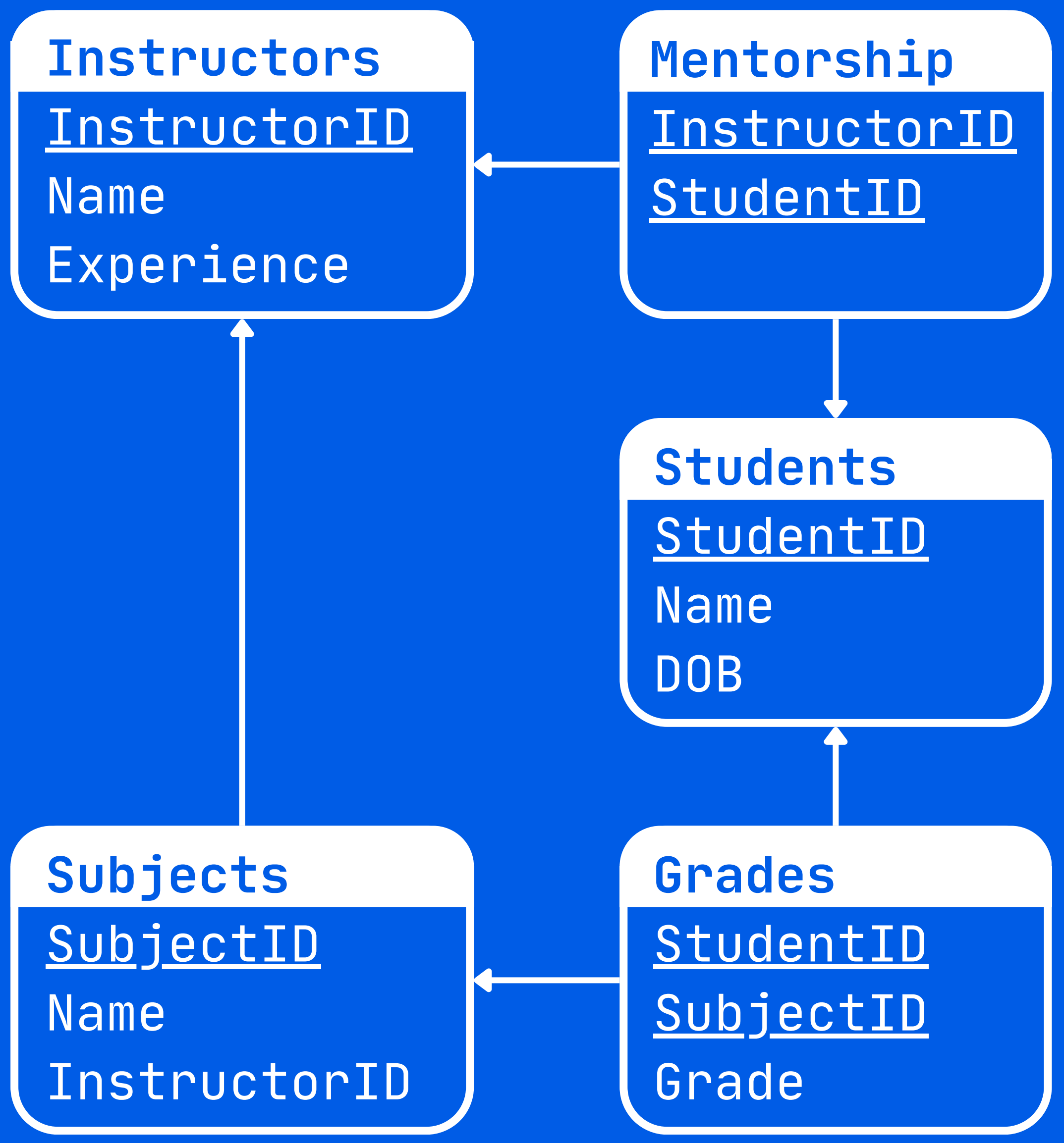
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

5 Find StudentID and their names of students who were born in or after year 2000.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

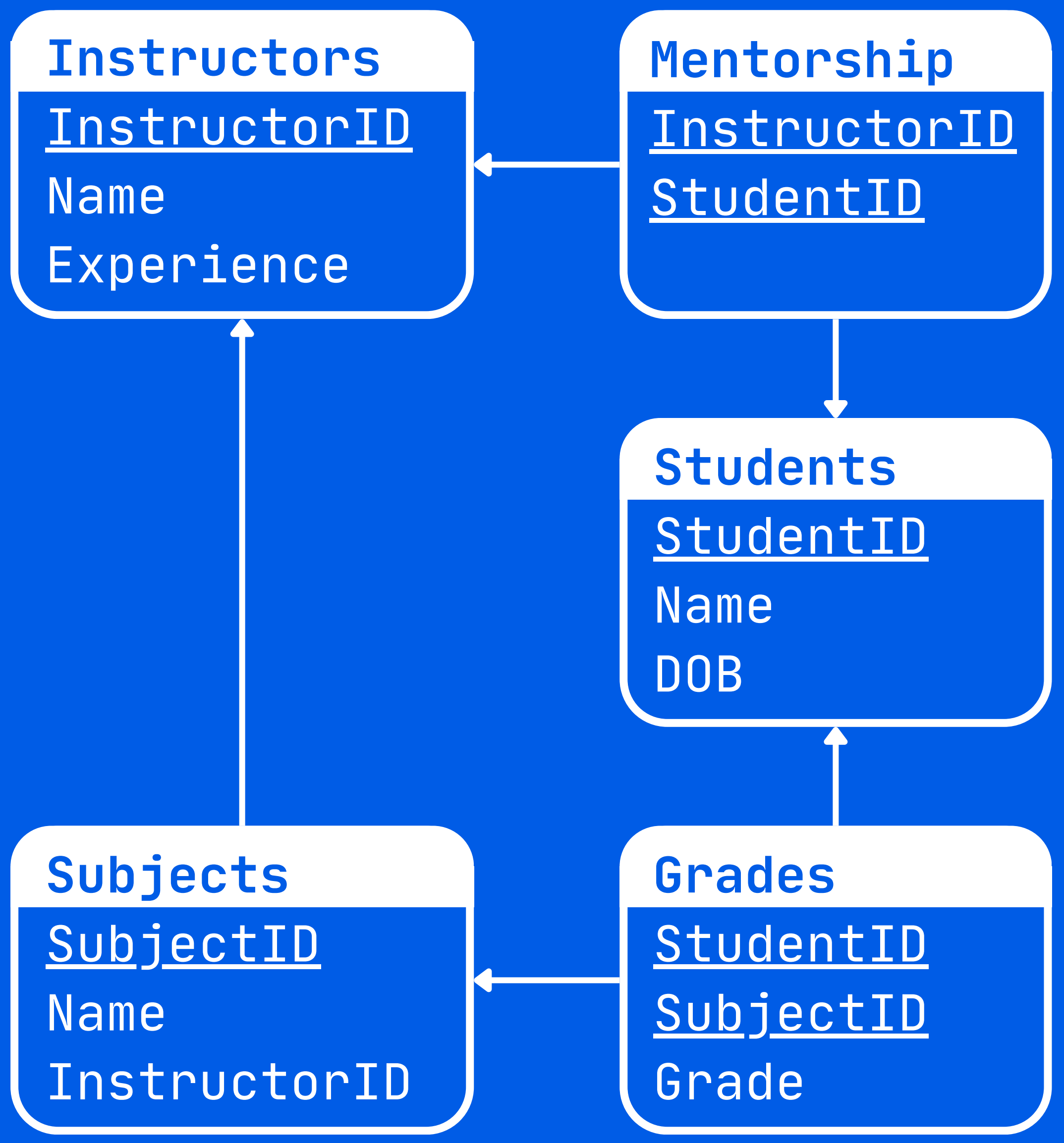
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

5 Find StudentID and their names of students who were born in or after year 2000.

{t.StudentID, t.Name | t ∈ *Students* ∧ t.DOB >= ‘2000-01-01’}



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

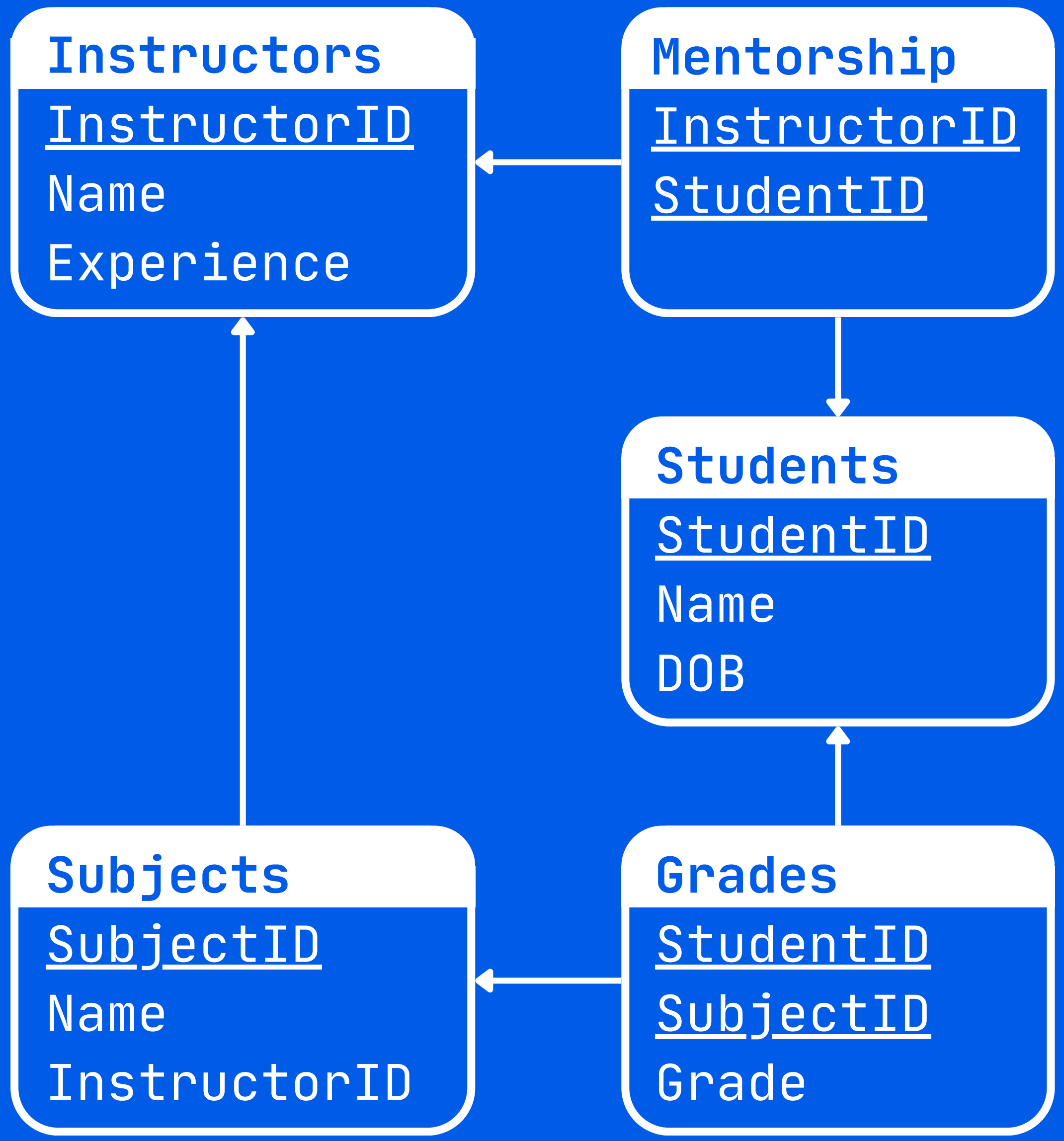
Mentorship: (InstructorID, StudentID)

TRC Queries

5 Find StudentID and their names of students who were born in or after year 2000.

```
{t.StudentID, t.Name | t ∈ Students ∧ t.DOB >=
'2000-01-01'}
```

6 Retrieve InstructorIDs along with the SubjectIDs of the subjects taught by those instructors who have more than 5 years of experience.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

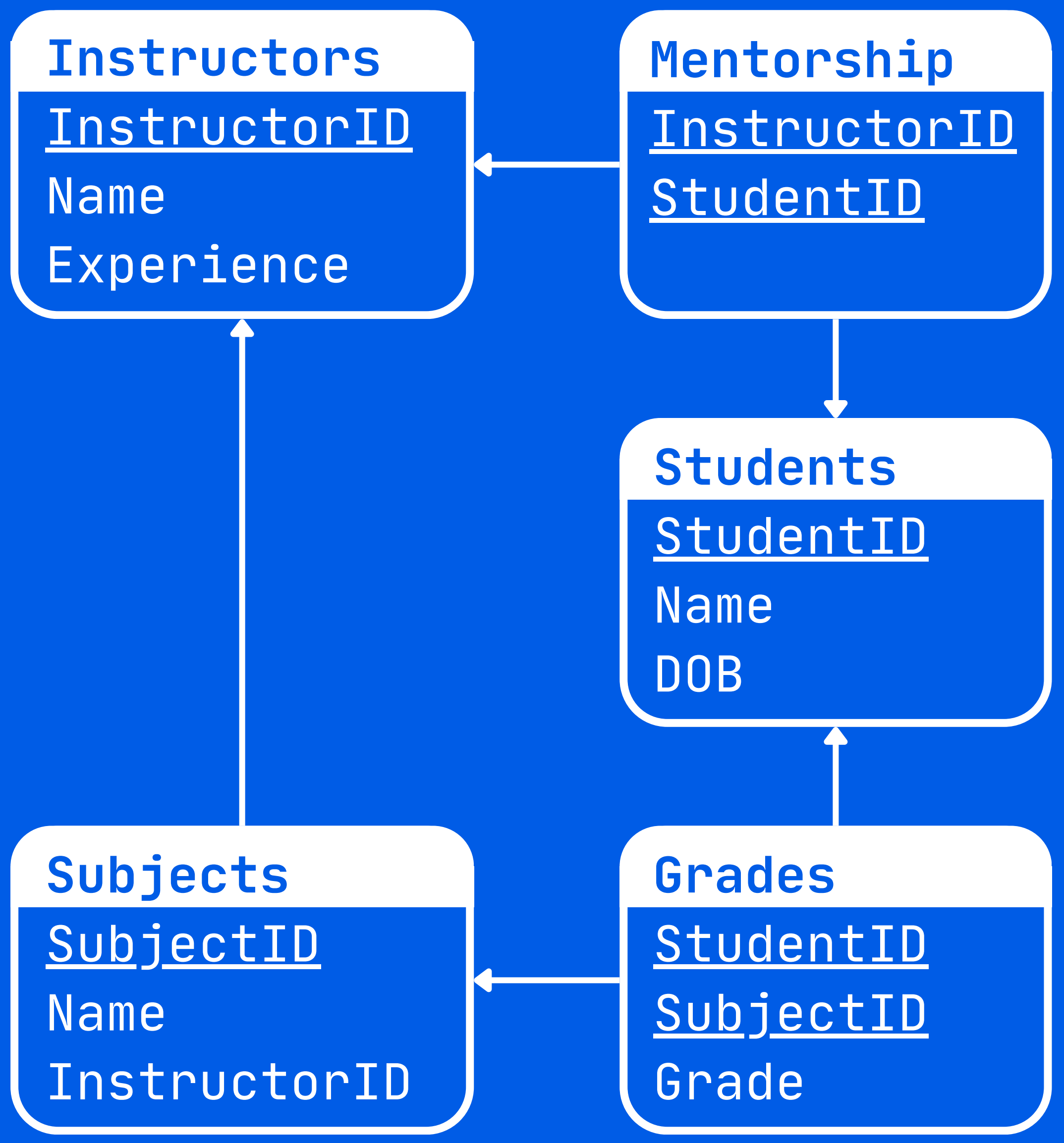
TRC Queries

5 Find StudentID and their names of students who were born in or after year 2000.

{t.StudentID, t.Name | t ∈ *Students* ∧ t.DOB >= ‘2000-01-01’}

6 Retrieve InstructorIDs along with the SubjectIDs of the subjects taught by those instructors who have more than 5 years of experience.

{t.InstructorID , t.SubjectID | t ∈ *Subjects* ∧ ∃ s (s ∈ *Instructors* ∧ s.InstructorID = t.InstructorID ∧ s.Experience > 5) }



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

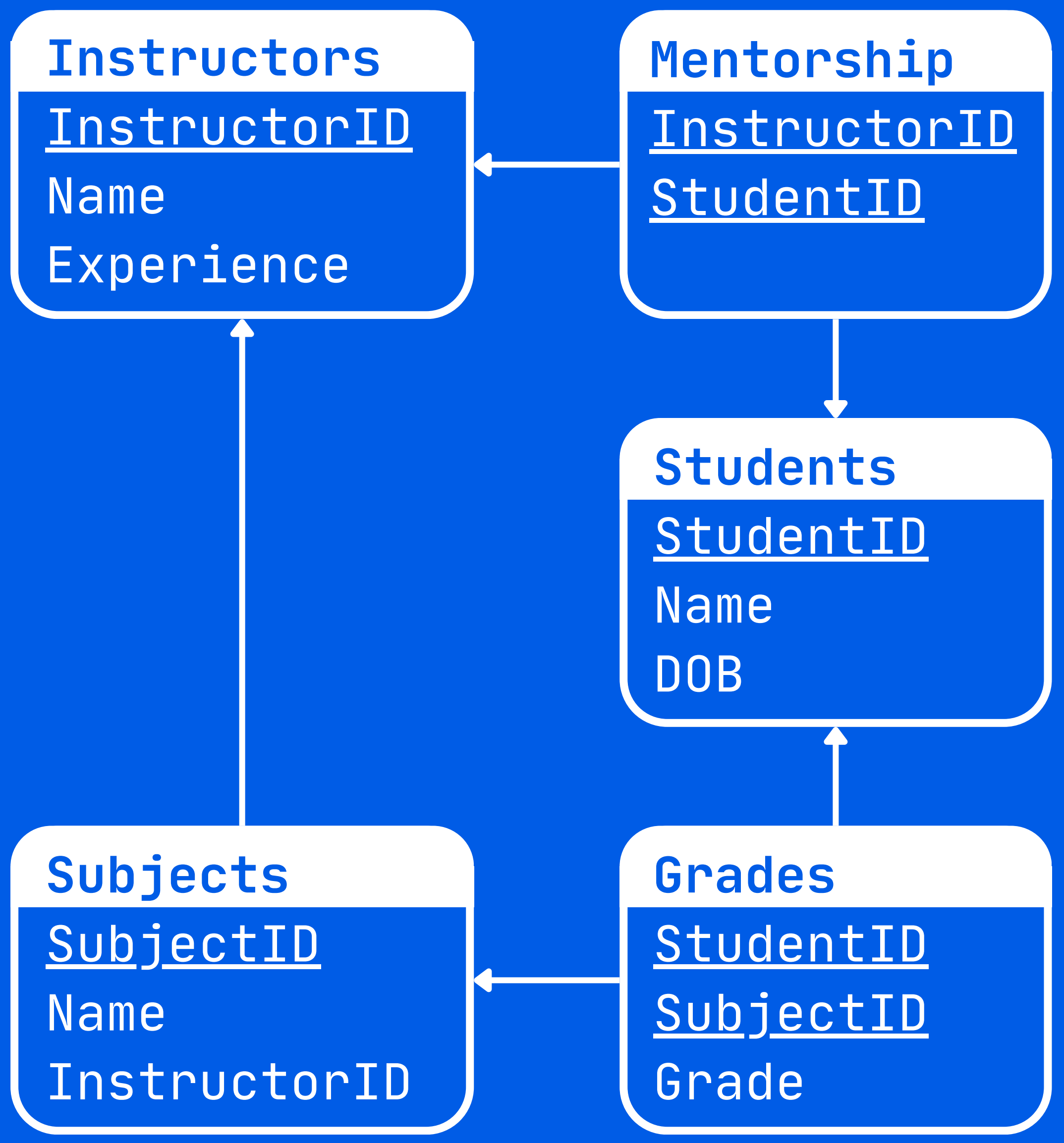
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

7 Retrieve InstructorID and name of instructors who mentor at least one student.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

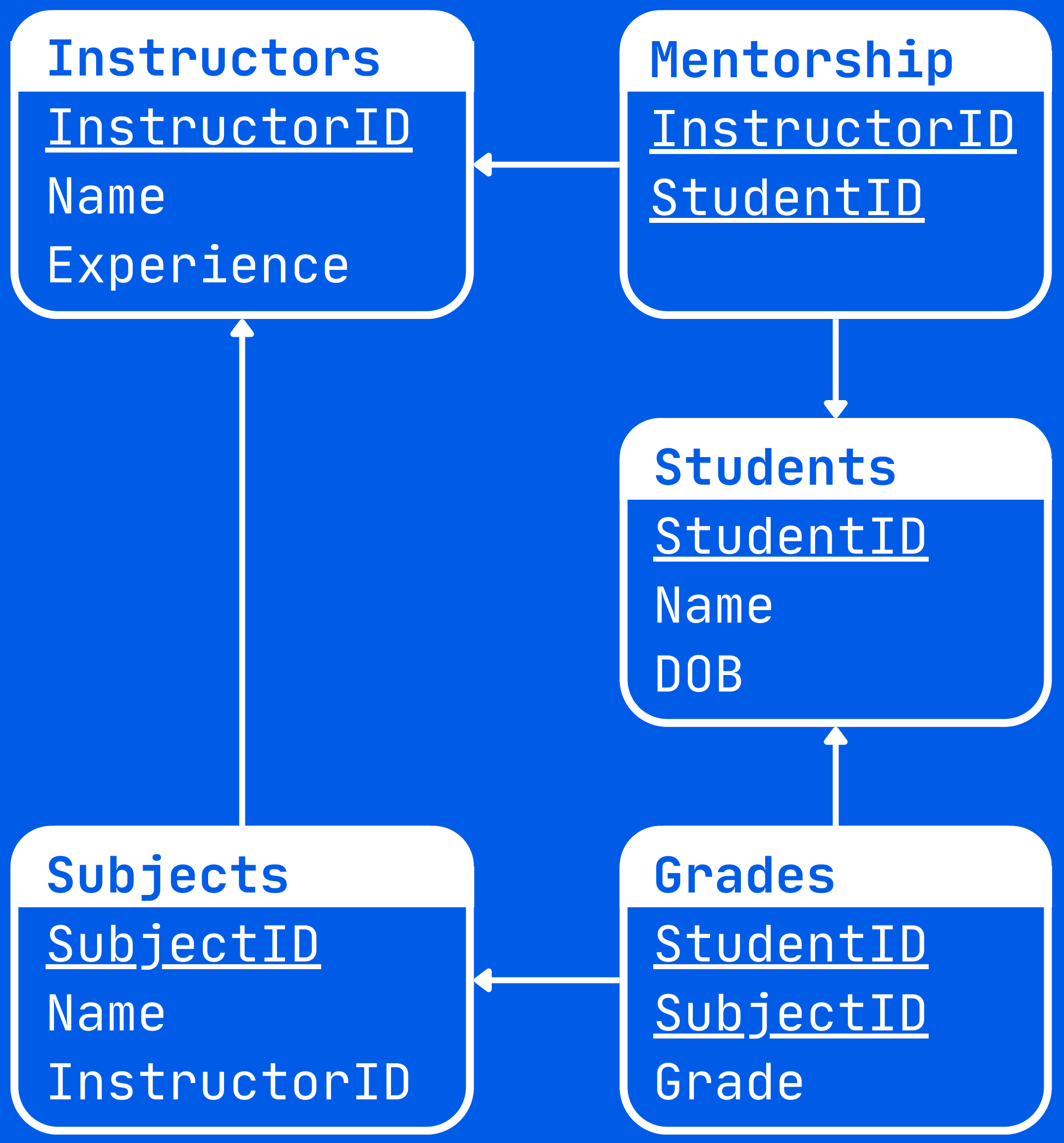
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

7 Retrieve InstructorID and name of instructors who mentor at least one student.

$\{ t.\mathrm{InstructorID} , t.\mathrm{Name} \mid t \in \mathit{Instructor} \land \exists s (s \in \mathit{Mentorship} \land s.\mathrm{InstructorID} = t.\mathrm{InstructorID}) \}$



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

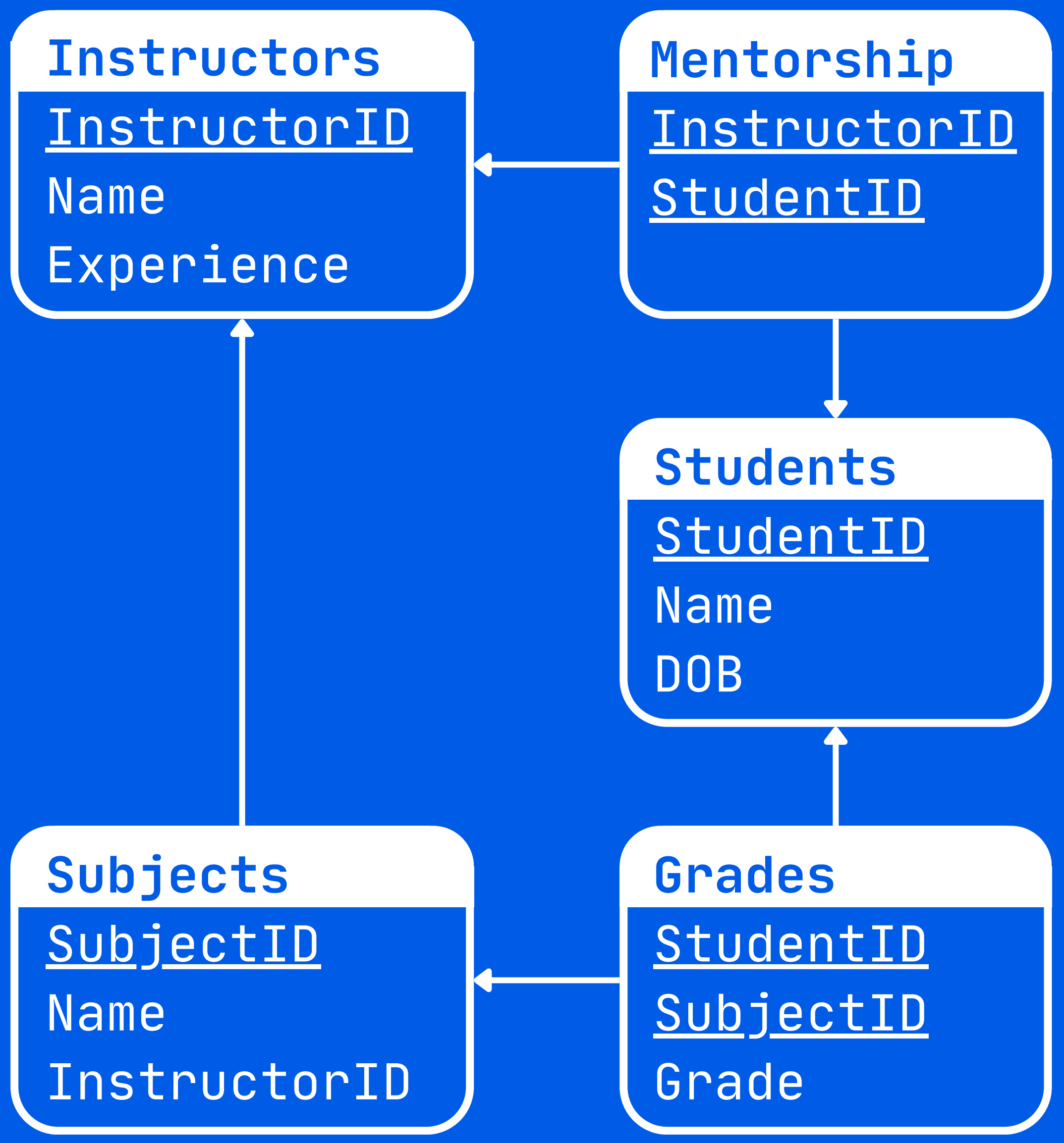
Mentorship: (InstructorID, StudentID)

TRC Queries

7 Retrieve InstructorID and name of instructors who mentor at least one student.

$$\{\ t.\mathrm{InstructorID}\ ,\ t.\mathrm{Name}\ \mid\ t \in \mathit{Instructor}\ \wedge\ \exists\ s\ (\ s \in \mathit{Mentorship}\ \wedge\ s.\mathrm{InstructorID} = t.\mathrm{InstructorID})\ \}$$

8 Retrieve StudentID and SubjectID for the students who have received grade “A” and are also mentored by an instructor.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

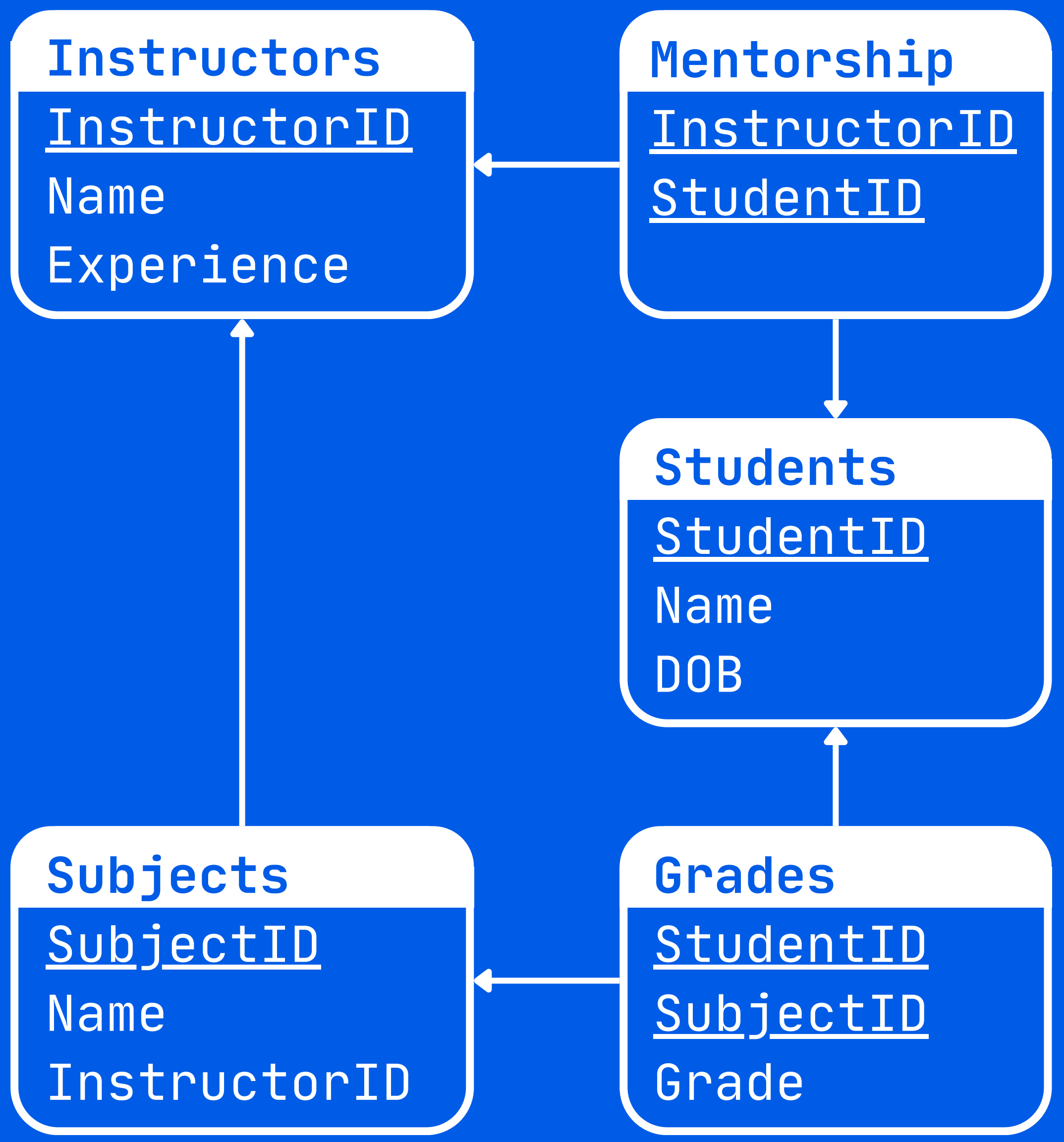
TRC Queries

7 Retrieve InstructorID and name of instructors who mentor at least one student.

{ t.InstructorID , t.Name | t ∈ *Instructor*
∧ ∃ s (s ∈ *Mentorship* ∧ s.InstructorID =
t.InstructorID) }

8 Retrieve StudentID and SubjectID for the students who have received grade “A” and are also mentored by an instructor.

**{t.StudentID,t.SubjectID | t ∈ *Grades* ∧
t.Grades = “A” ∧ ∃ s (s ∈ *Mentorship* ∧
t.StudentID = s.StudentID) }**



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

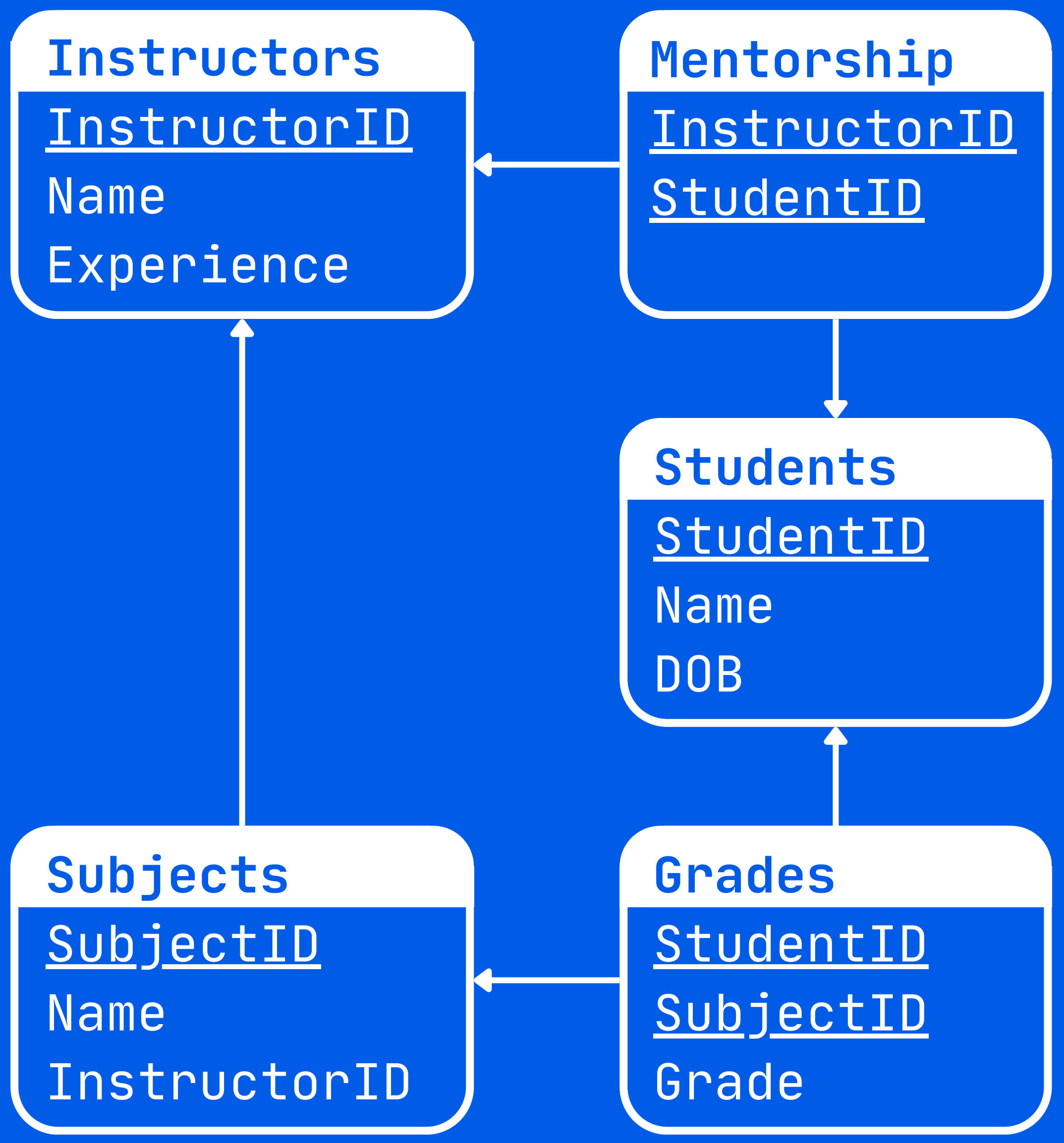
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

9 List the details of the students whose DOB is of before year 2000 or got grade “A”.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

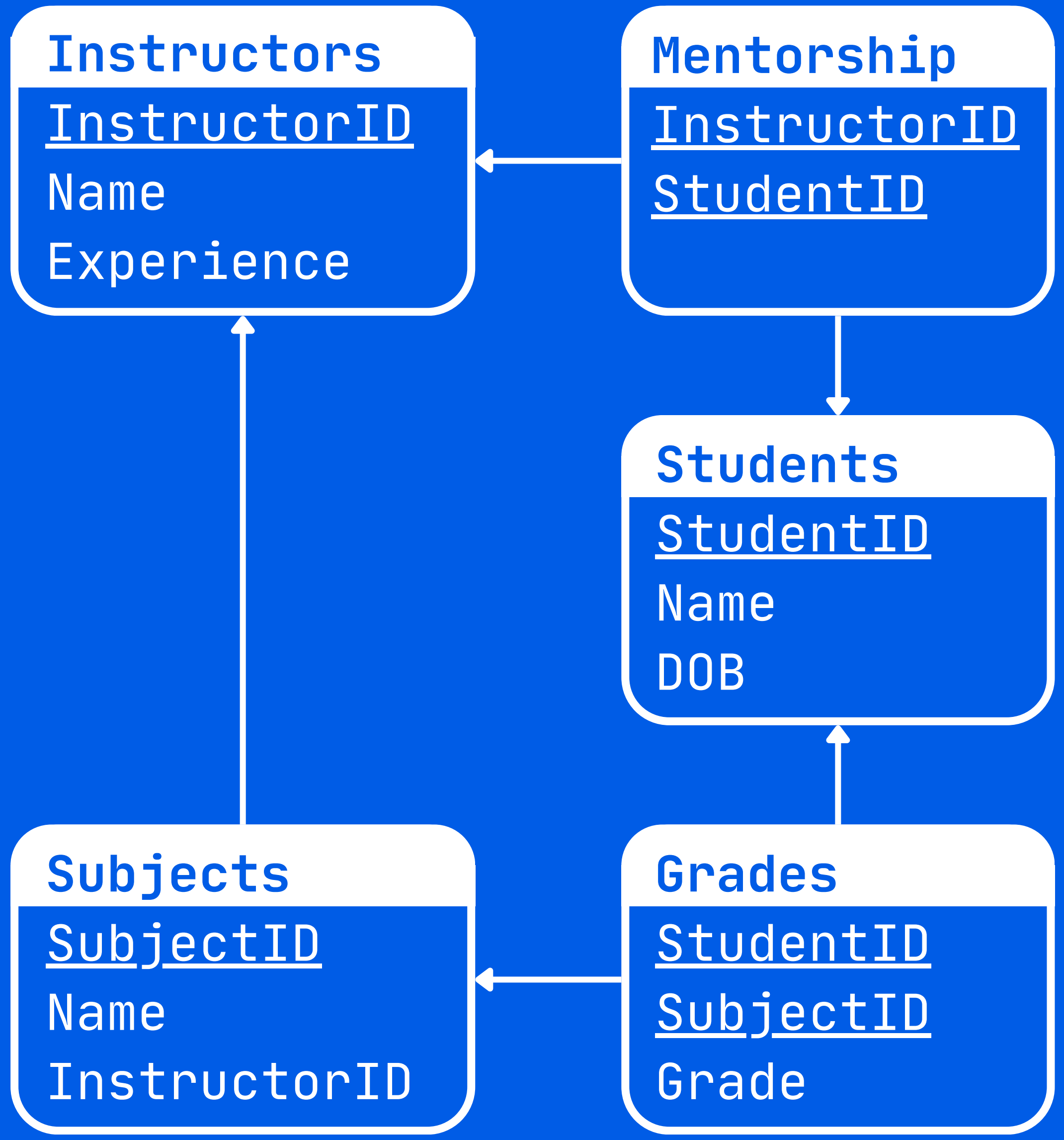
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

9 List the details of the students whose DOB is of before year 2000 or got grade “A”.

{t | t ∈ *Students* ∧ t.DOB < ‘2000-01-01’ ∨ ∃s(s ∈ *Grades* ∧ s.Grade = “A” ∧ s.StudentID = t.StudentID)}



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

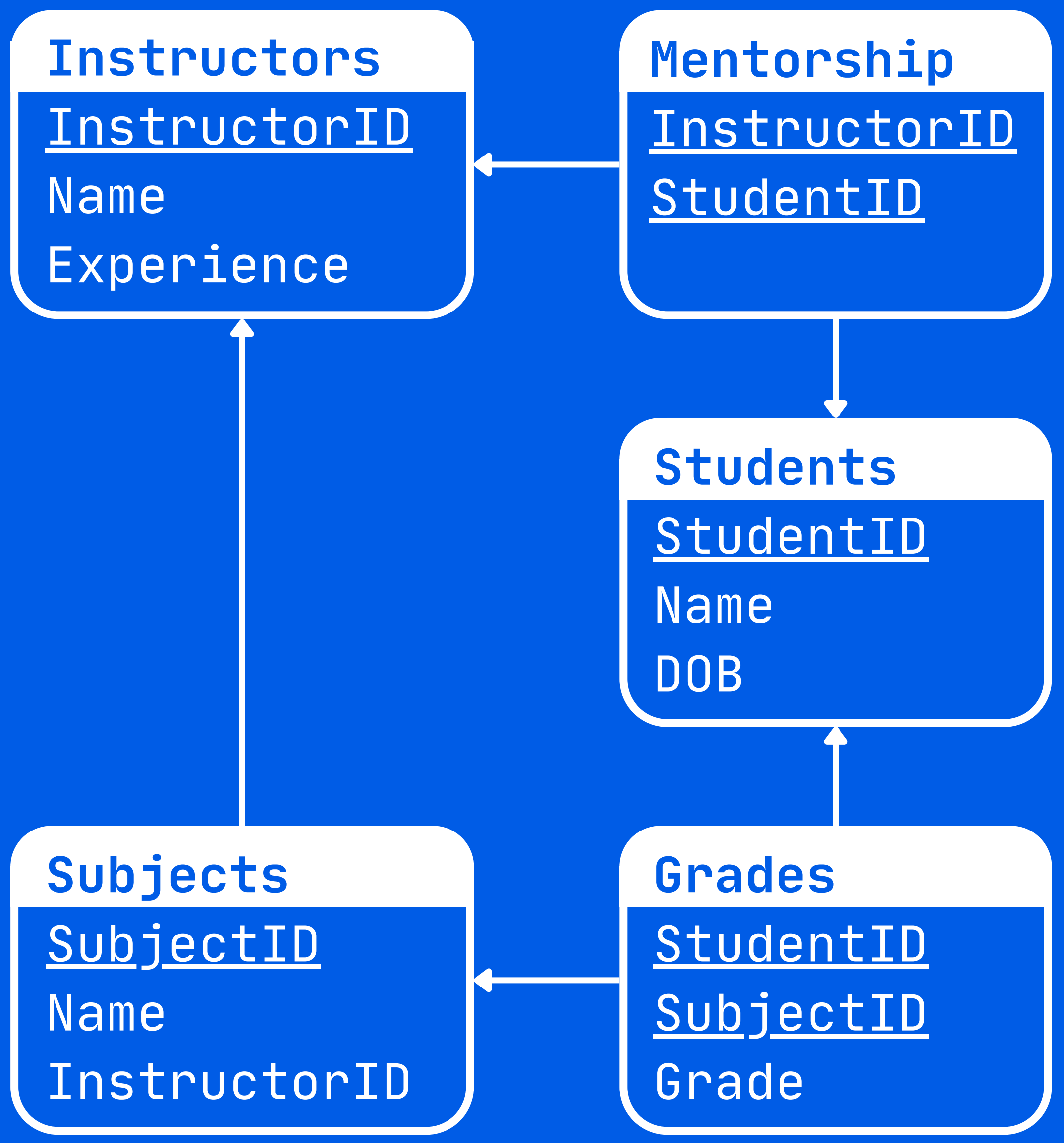
Mentorship: (InstructorID, StudentID)

TRC Queries

9 List the details of the students whose DOB is of before year 2000 or got grade “A”.

$$\{t \mid t \in \textit{Students} \land t.\text{DOB} < \text{‘2000-01-01’} \lor \exists s(\, s \in \textit{Grades} \land s.\text{Grade} = \text{“A”} \land s.\text{StudentID} = t.\text{StudentID})\}$$

10 Find the names of students who have taken a subject taught by 'Korth’.



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

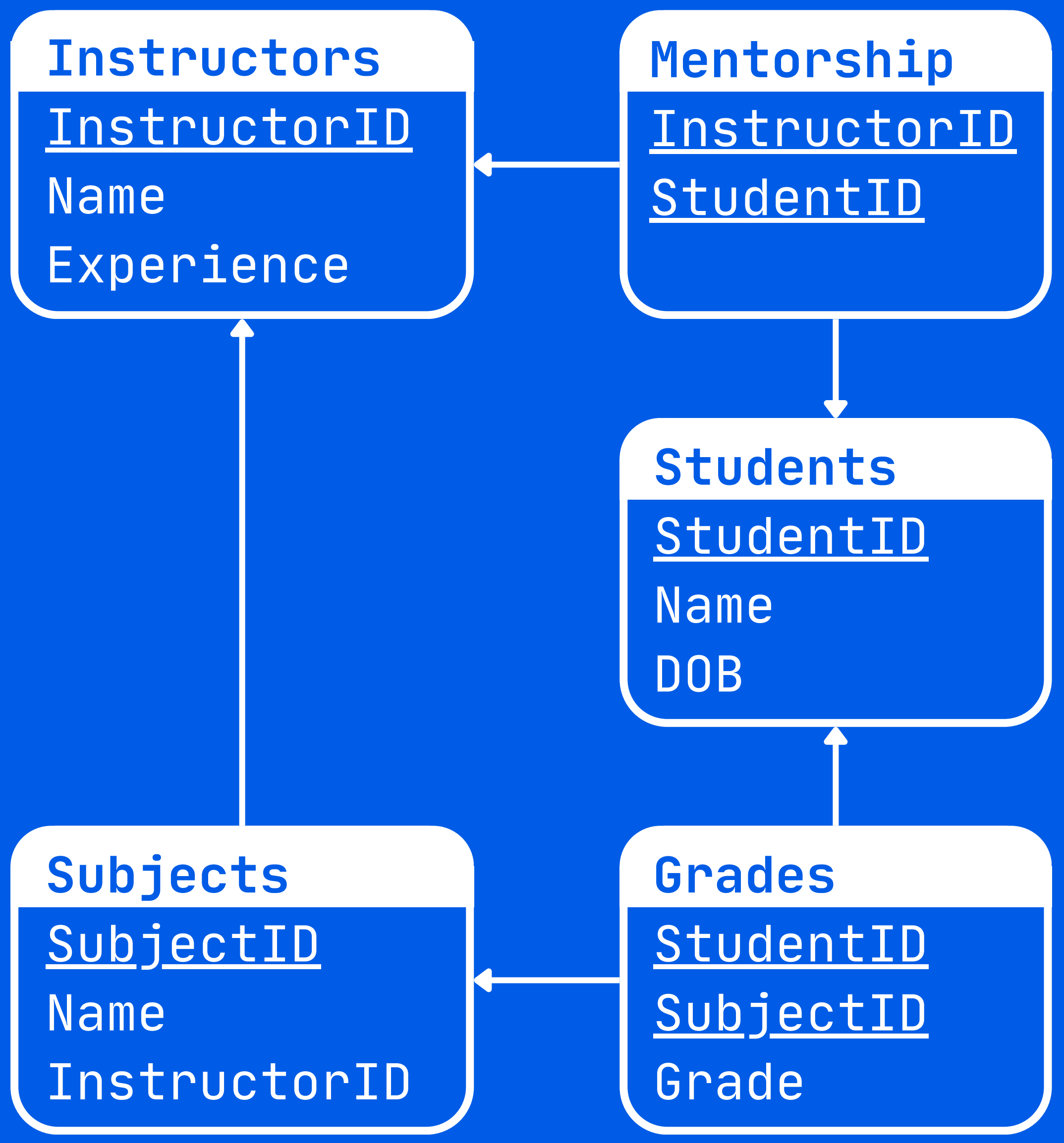
TRC Queries

9 List the details of the students whose DOB is of before year 2000 or got grade “A”.

{t | t ∈ *Students* ∧ t.DOB < ‘2000-01-01’ ∨ ∃s(s ∈ *Grades* ∧ s.Grade = “A” ∧ s.StudentID = t.StudentID)}

10 Find the names of students who have taken a subject taught by 'Korth’.

{t.name | t ∈ *Students* ∧ ∃g(g∈*Grades* ∧ t.StudentID=g.StudentID ∧ ∃s(s∈*Subjects* ∧ g.SubjectID=s.SubjectID ∧ ∃i(i∈*Instructors* ∧ i.InstructorID=s.InstructorID ∧ i.Name = “Korth”))) }



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

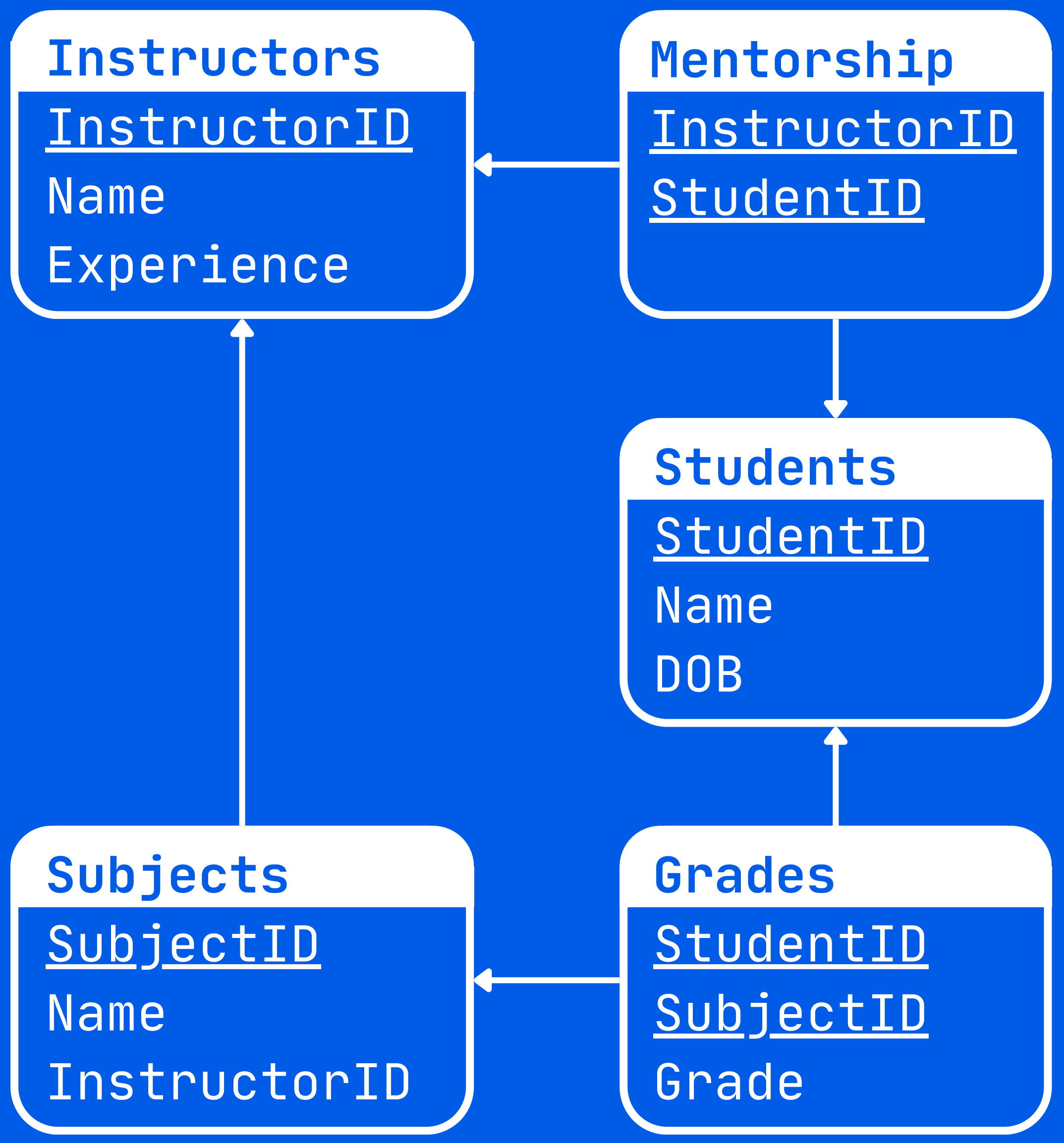
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

11 Retrieve instructor name and student name for each pair of instructor and student from mentorship table.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

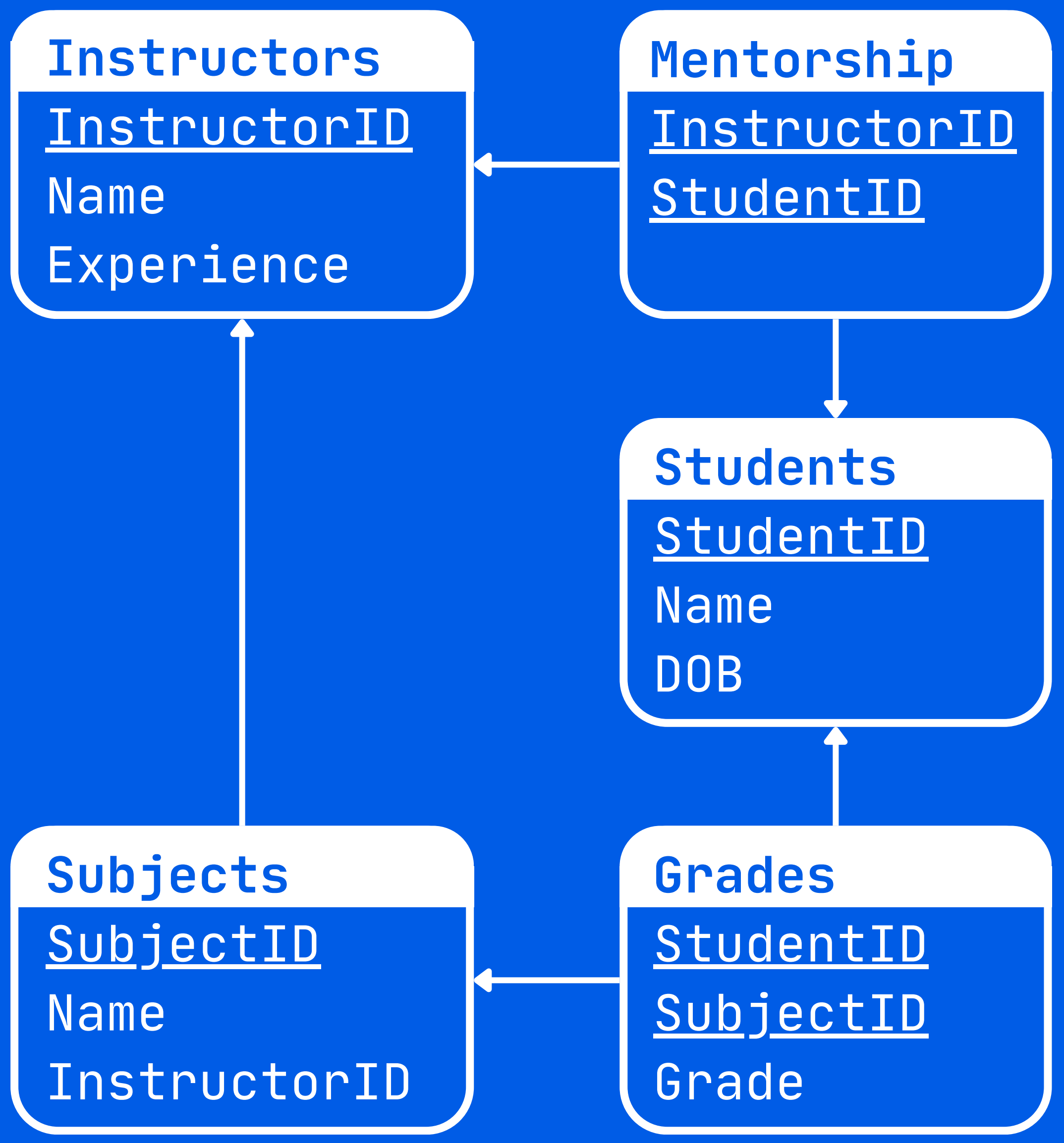
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC Queries

11 Retrieve instructor name and student name for each pair of instructor and student from mentorship table.

$$\{ t \mid \exists i(i \in Instructors \land \exists s(s \in Students \land \exists m(m \in Mentorship \land i.InstructorID = m.InstructorID \land s.StudentID = m.StudentID \land t.InstructorName = i.Name \land t.StudentName = s.Name)))\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

TRC V/S SQL

TRC	SQL
Theoretical query language used to describe what data to fetch	**Practical** and industry-standard query language
Based on mathematical logic (**predicate logic**)	Based on **relational calculus**, but **extended** with many features
Primarily used in **academic** and **theoretical contexts**	Widely used in **real-world applications** and **DBMSs**
Tuple relational calculus focuses only on **data retrieval logic**	Supports **data retrieval**, **manipulation**, and **control**
No user interface or tools, exists as **conceptual model**.	Supported by a vast ecosystem of **tools**, **extensions**, and **GUIs**
Eg: $\{t \mid t \in \textit{Students}\}$	`SELECT * FROM Students;`

Domain Relational Calculus (DRC)

Selecting **attributes** in a relation that satisfy a given condition (or predicate).

- Working same as that of TRC, except we are selecting **columns**.
- Basic **syntax**: $\{ <a_1, a_2, \ldots a_n> \mid P(a_1, a_2, \ldots a_n) \}$

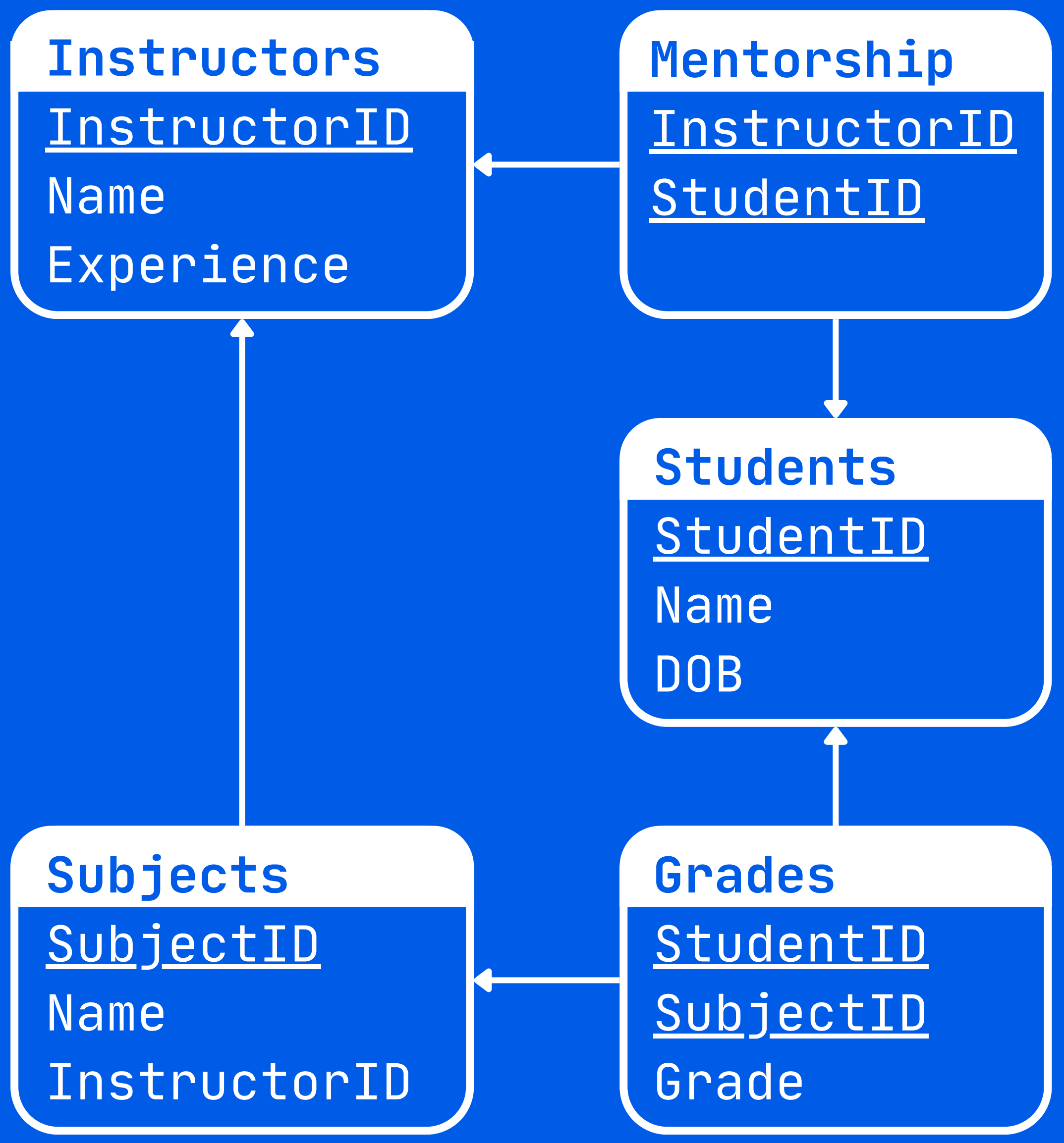
Result
(Domain variables ‘a_i’)

Predicate
(Condition used to fetch attributes ‘a_i’)

- **Result:** Set of attributes **a_1**, **a_2**,...**a_n** such that predicate ‘**P**’ is true for attributes **a_1**, **a_2**,...**a_n**.
- **Notations:**
  - **$<a_1 , a_2 , \ldots a_n > \in$ r:** Where **r** is a relation on n attributes and a_1 , a_2 ,...a_n are domain variables/constants.
  - **P:** predicate i.e, the condition on tuples

DRC Queries

1 List the names of all students in Students relation.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

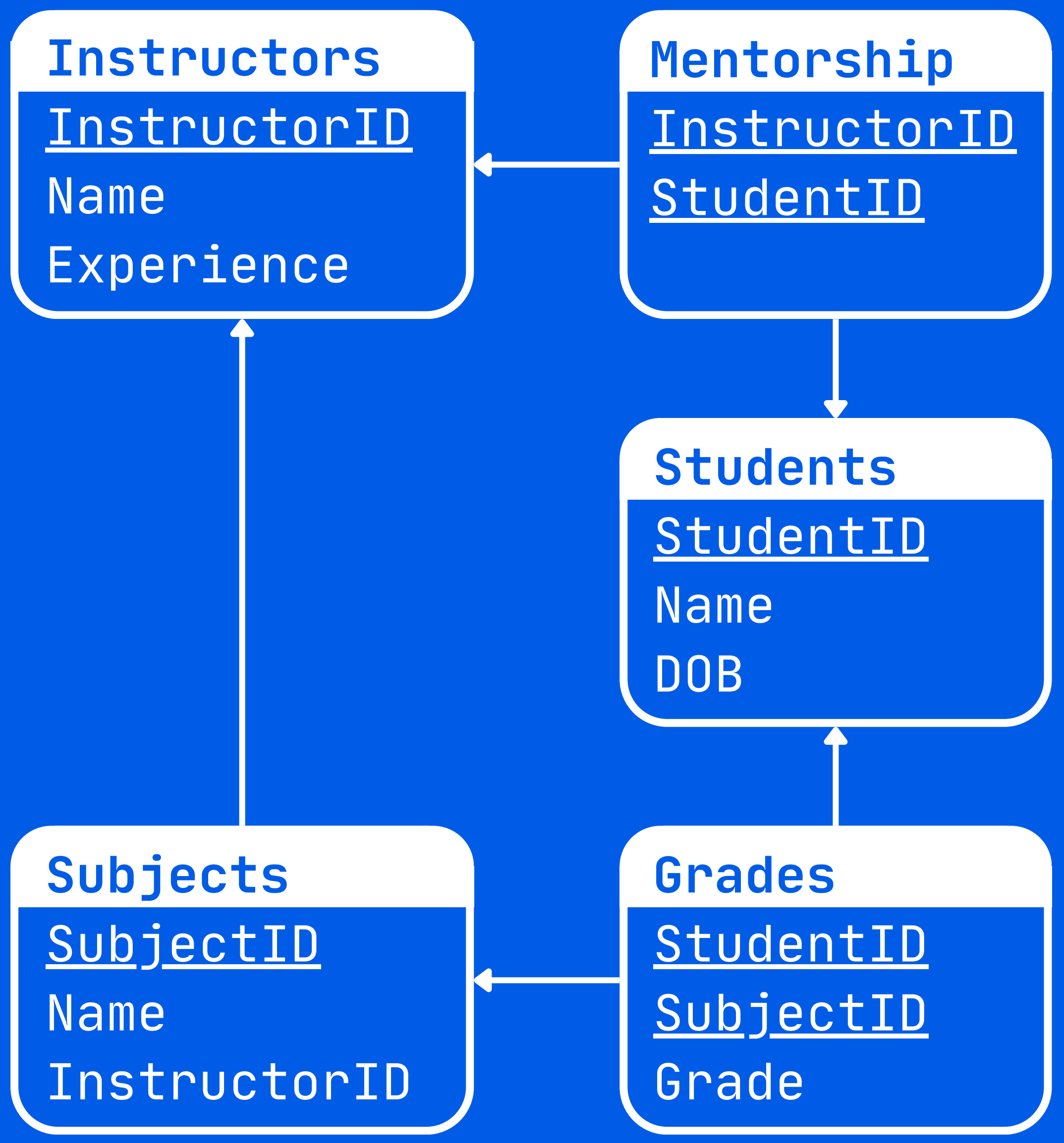
Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

DRC Queries

1 List the names of all students in Students relation.

$$\{ n \mid (\exists r)(\exists d)(\mathit{Students}(r, n, d))\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

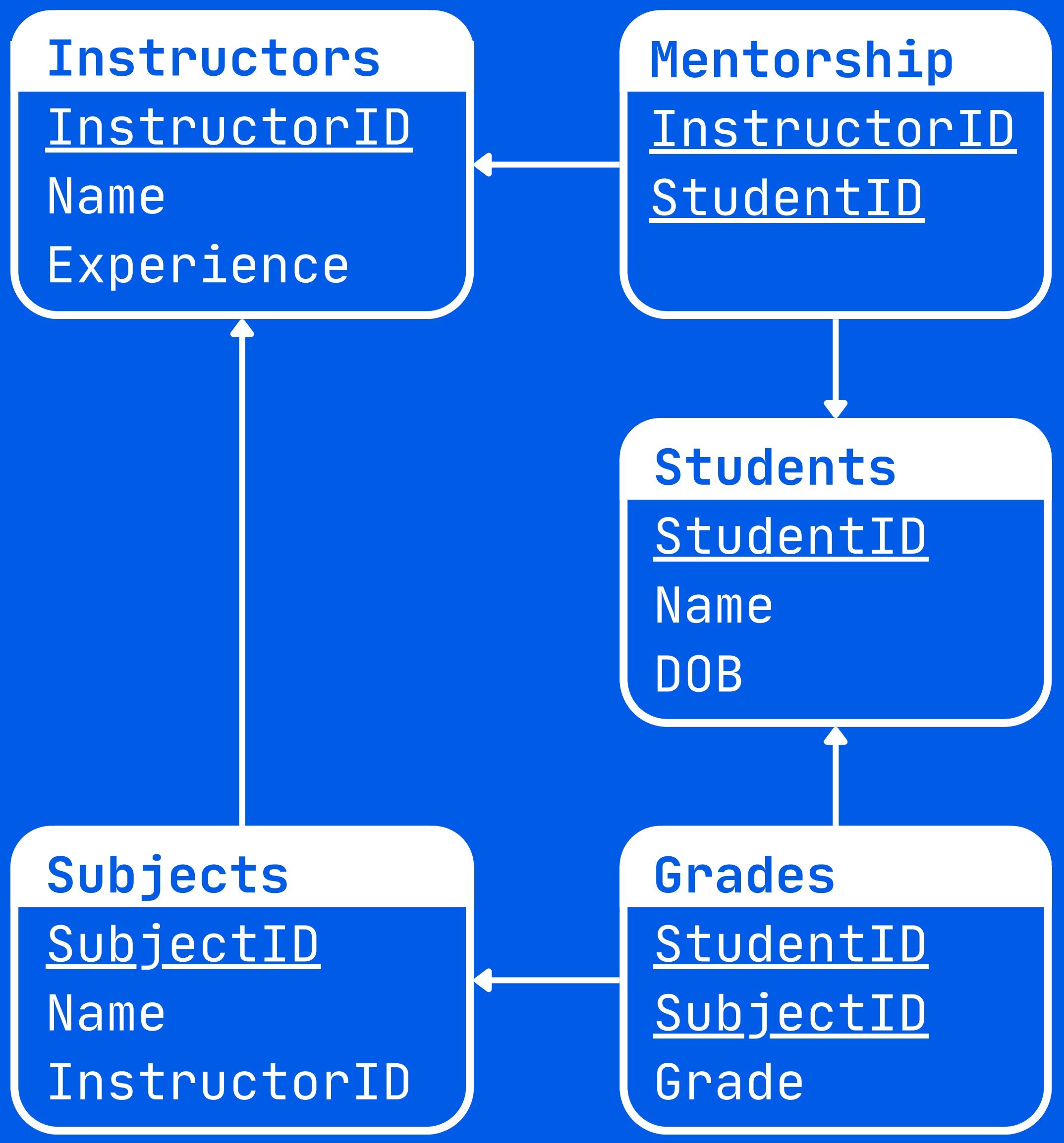
Mentorship: (InstructorID, StudentID)

DRC Queries

1 List the names of all students in Students relation.

$$\{ n \mid (\exists r)(\exists d)(\mathit{Students}(r, n, d))\}$$

2 Find names of instructors with more than 10 years of experience.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

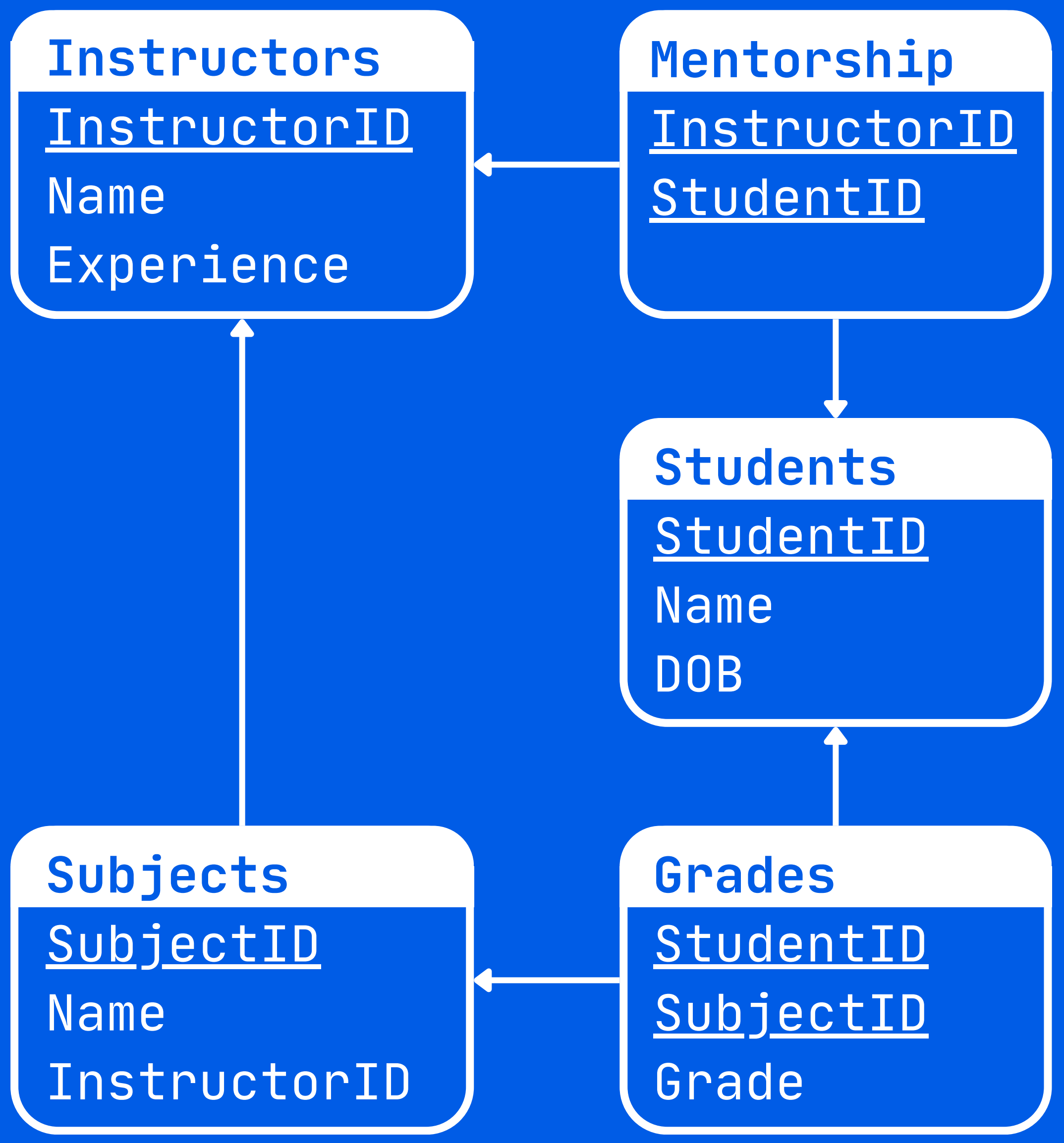
DRC Queries

1 List the names of all students in Students relation.

$$\{\ n \mid (\exists r)(\exists d)(\mathit{Students}(r,\ n,\ d))\}$$

2 Find names of instructors with more than 10 years of experience.

$$\{\ x \mid (\exists i)(\exists e)(\mathit{Instructors}(i,\ x,\ e) \land e > 10)\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

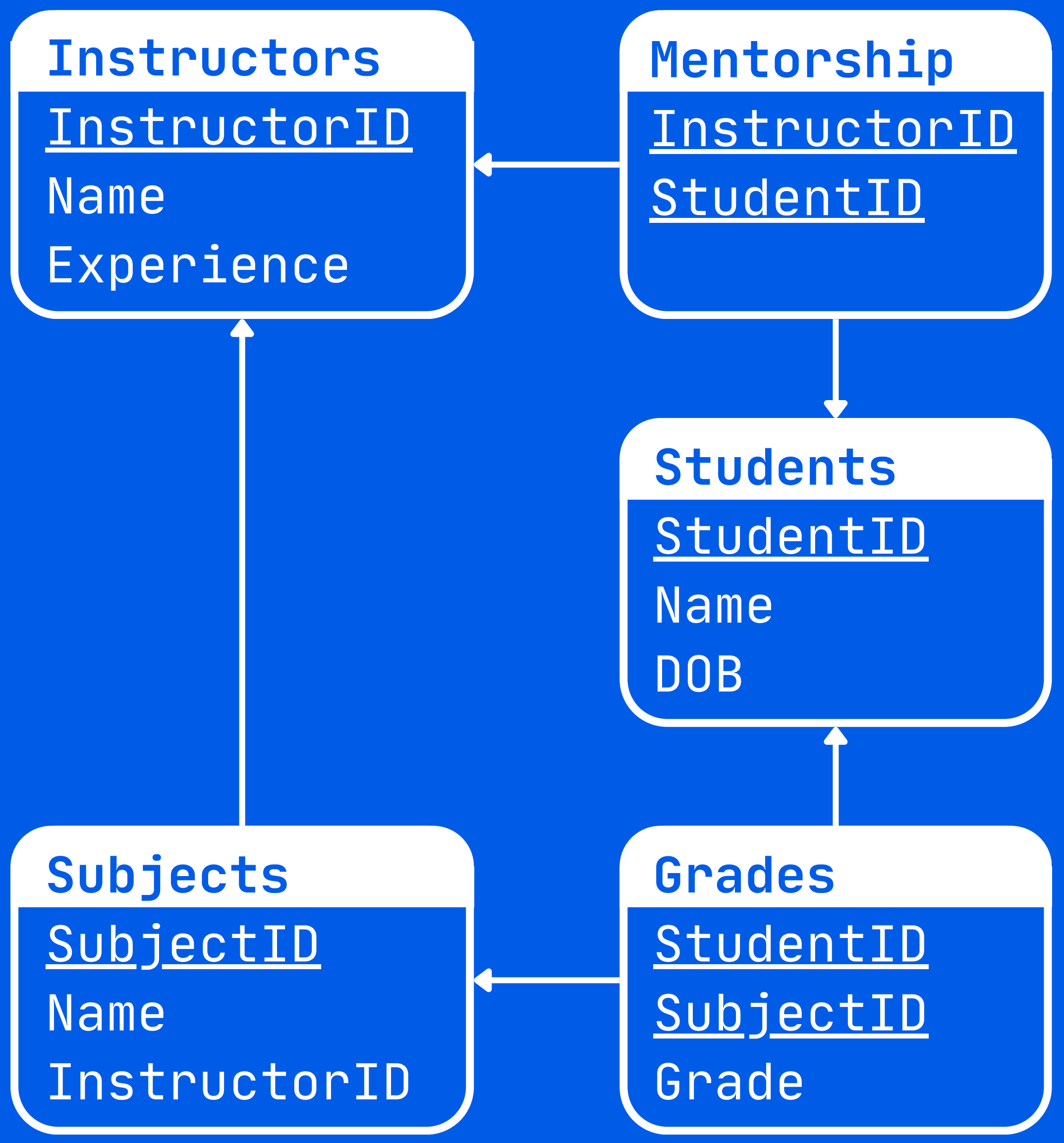
Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

DRC Queries

3 List names of students who got an A grade in any subject.



Students: (<u>StudentID</u>, Name, DOB)

Instructors: (<u>InstructorID</u>,Name,Experience)

Subjects: (<u>SubjectID</u>, Name, InstructorID)

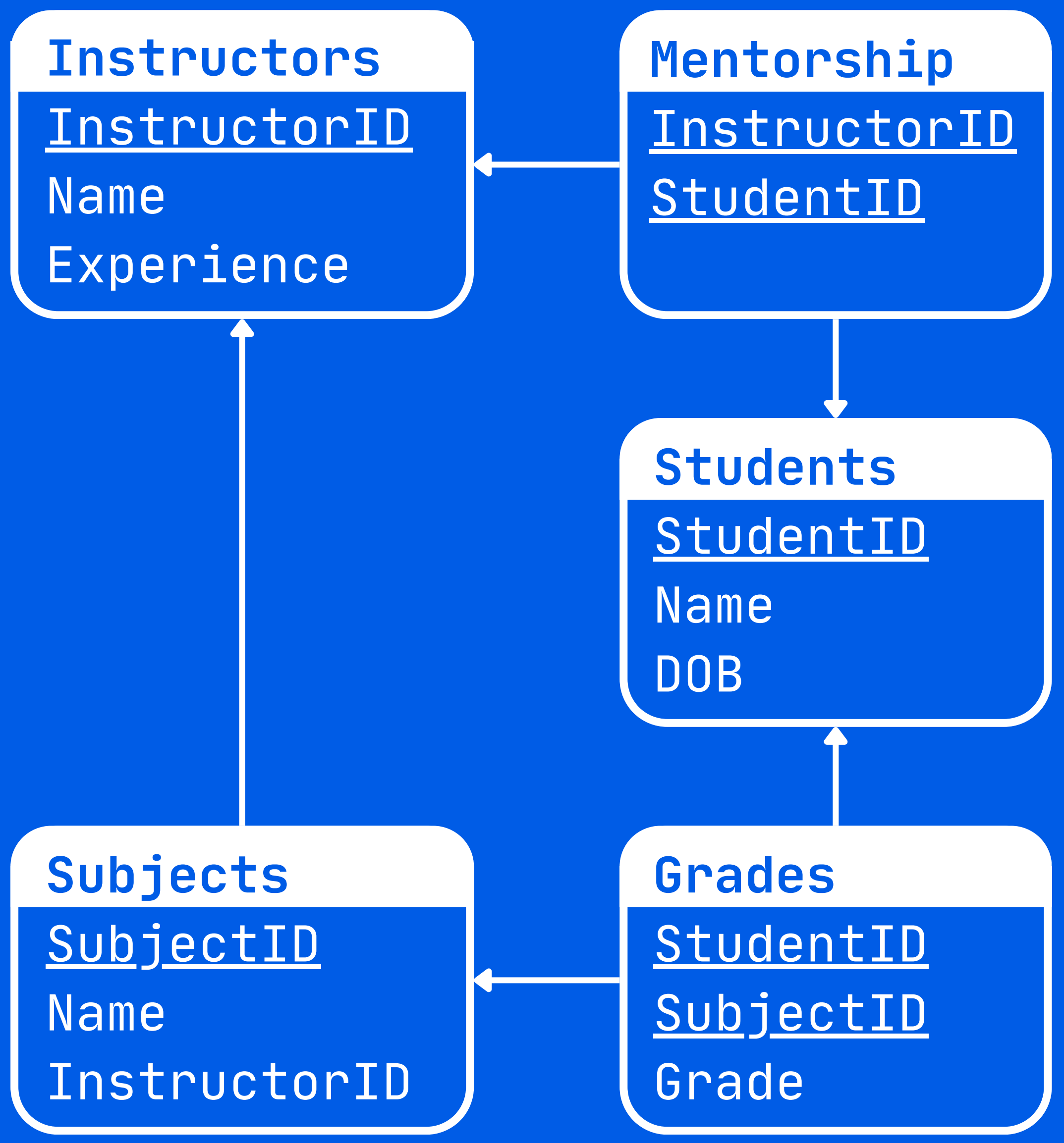
Grades: (<u>StudentID</u>, <u>SubjectID</u>, Grade)

Mentorship: (<u>InstructorID</u>, <u>StudentID</u>)

DRC Queries

3 List names of students who got an A grade in any subject.

$$\{\ n \mid (\exists r)(\exists d)(\exists s)(\exists g)(\mathit{Students}(r, n, d) \land \mathit{Grades}(r, s, g) \land g='A')\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

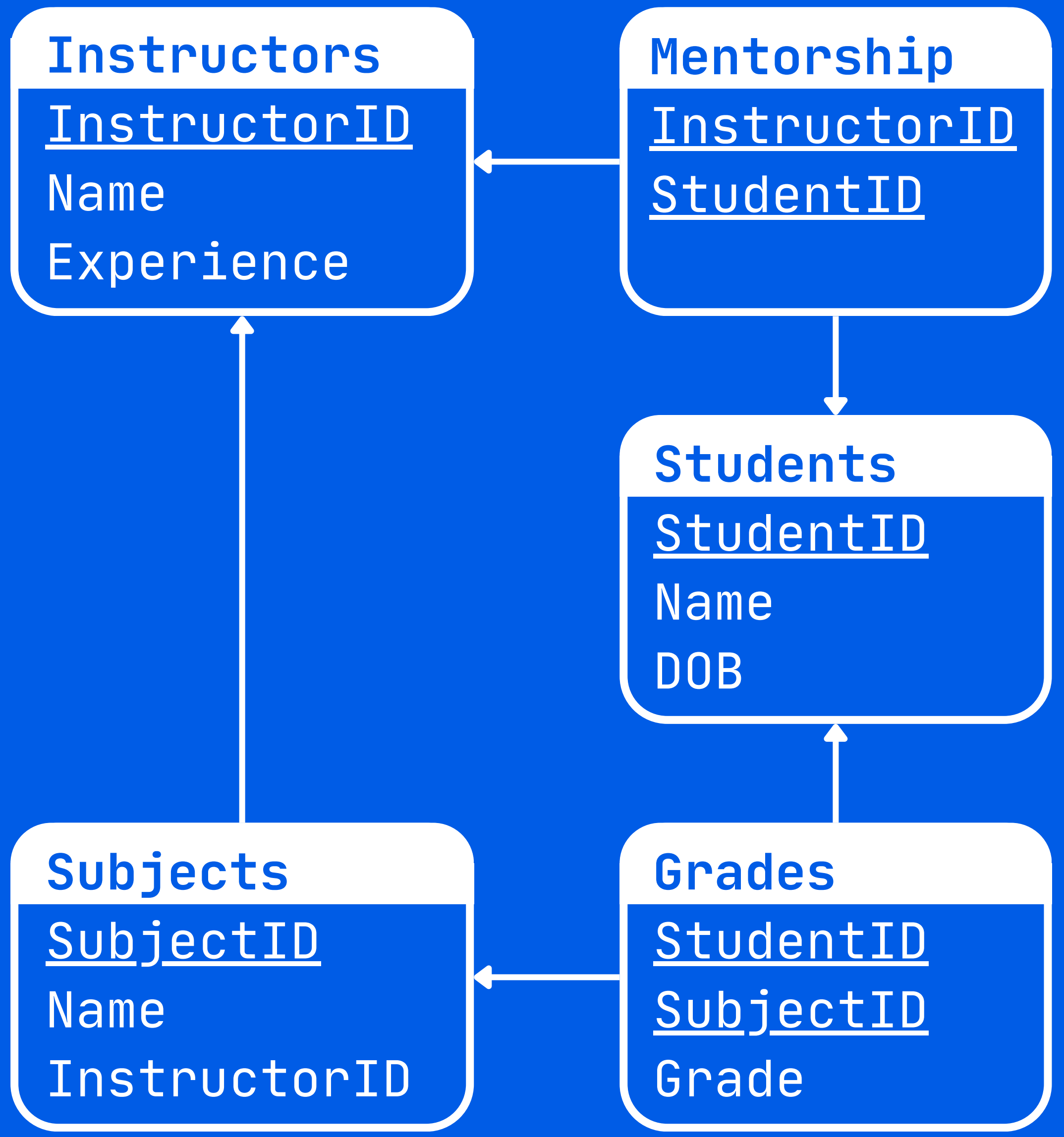
Mentorship: (InstructorID, StudentID)

DRC Queries

3 List names of students who got an A grade in any subject.

$$\{ n \mid (\exists r)(\exists d)(\exists s)(\exists g)(\mathit{Students}(r, n, d) \land \mathit{Grades}(r, s, g) \land g='A')\}$$

4 List student names mentored by instructors with over 15 years of experience.



Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

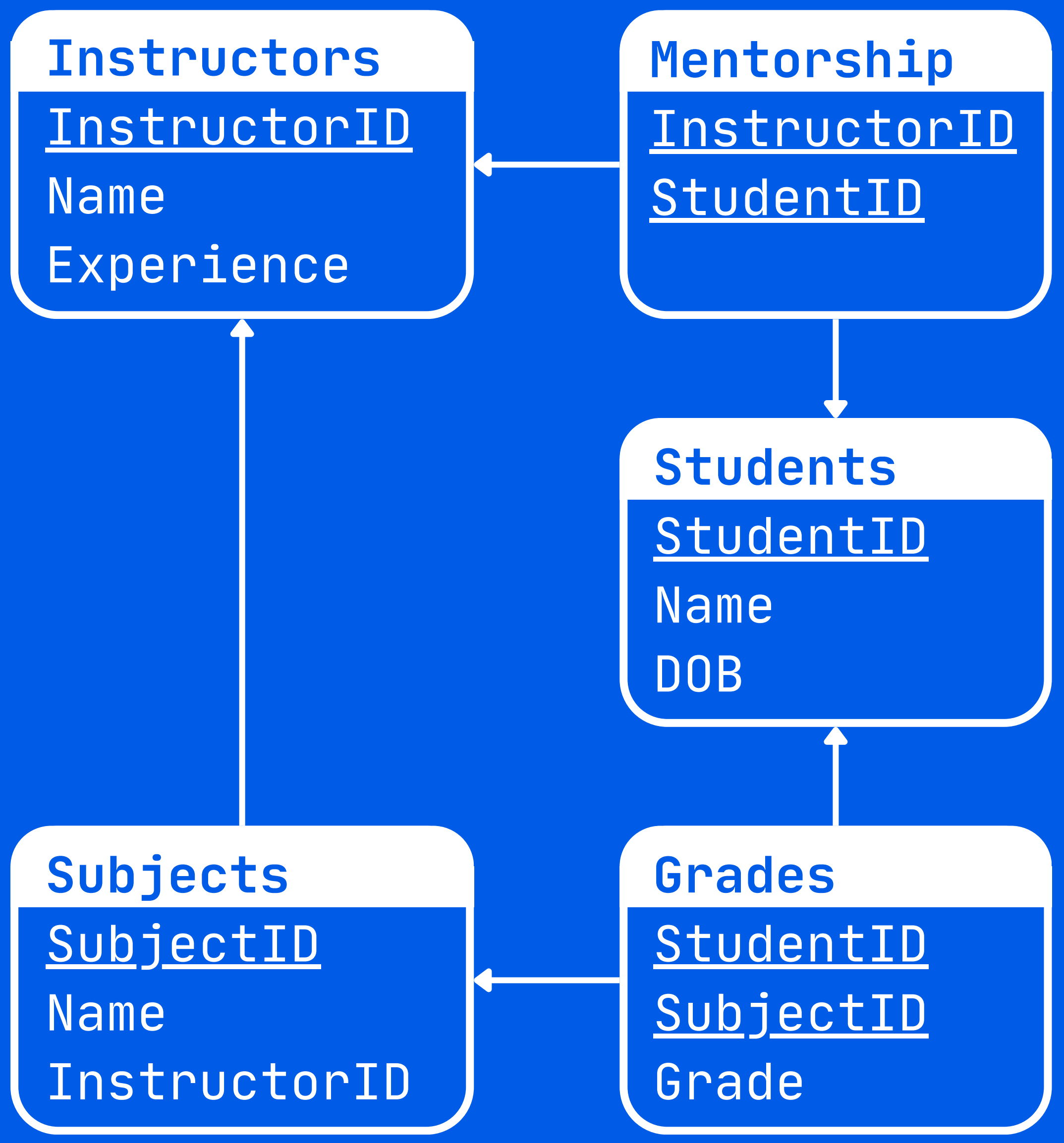
DRC Queries

3 List names of students who got an A grade in any subject.

$$\{ n \mid (\exists r)(\exists d)(\exists s)(\exists g)(\mathit{Students}(r, n, d) \wedge \mathit{Grades}(r, s, g) \wedge g='A')\}$$

4 List student names mentored by instructors with over 15 years of experience.

$$\{ n \mid (\exists r)(\exists d)(\exists i)(\exists x)(\exists e)(\mathit{Students}(r, n, d) \wedge \mathit{Mentorship}(i,r) \wedge \mathit{Instructors}(i, x, e) \wedge e>15)\}$$



Instructors
InstructorID
Name
Experience

Mentorship
InstructorID
StudentID

Students
StudentID
Name
DOB

Subjects
SubjectID
Name
InstructorID

Grades
StudentID
SubjectID
Grade

Students: (StudentID, Name, DOB)

Instructors: (InstructorID,Name,Experience)

Subjects: (SubjectID, Name, InstructorID)

Grades: (StudentID, SubjectID, Grade)

Mentorship: (InstructorID, StudentID)

V/S

TRC	DRC
Based on **tuples as variables**, each representing an entire row	Based on **attribute-level variables**, each representing a value
Queries are expressed in terms of **relations** and **tuples**	Queries are expressed using **individual domain values**
Queries tend to be **shorter** and **cleaner** for multi-attribute relations	Queries can become **verbose** as each field is explicitly named
Better for understanding **row-based query formulation**	Better for understanding **column-level constraints**
Often preferred for **practical query formulation**	Often used in **academic** and **logic-based proofs**
Eg: $\{ t.Name,t.DOB \mid t\in \mathit{Students}\}$	$\{<n, d> \mid \exists i(Students(n,d,i))\}$

What’s wrong with the below query?

$$\{t \mid \neg(t \in student)\}$$

What’s **wrong** with the below query?

$$\{t \mid \neg(t \in student)\}$$

∵ There are infinitely many tuples that are not in *student.*

∴ This leads to a result with **infinite** tuples.

Such queries are known as **Unsafe Queries**

Safe Expressions in Relational Calculus

A query is safe if it produces a **finite** result set.

- A safe expression in Relational Calculus ensures that the result of a query is **finite** and **computable**.
- It avoids returning an infinite number of tuples, which is essential for **practical database execution**.
- Guarantees that results come from **actual**, **existing data** in the database and not hypothetical or undefined values.

Safe Expressions in Relational Calculus

How to ensure **Safety of Expressions**?

1. **Restrict variables to finite domains:** Every variable must range over a finite relation (i.e., a known table), not the entire universal domain.
2. **Use only values from the database or constants:** Ensure all values in the query come from existing data or explicit constants, not hypothetical or infinite values.
3. **Apply negation and quantifiers within restricted domains:** Use ¬ (NOT) and ∀ (FOR ALL) only when the scope is limited to a defined relation, avoiding unbounded results.

Safe version of the previous query!

$$\{t \mid \underbrace{t \in Person}_{} \wedge \neg(t \in Student)\}$$

∵ It restricts the domain to known values from the *Person* relation.

∴ This leads to a result with **finite** tuples.

Leading to a **Safe Query**

Summary

- **Relational Calculus:** Non-procedural query language that focuses on what to retrieve, not how to retrieve it.
- Based on **Predicate Calculus**, using logical expressions, variables, and quantifiers.
- **Two types:** Tuple Relational Calculus (**TRC**) and Domain Relational Calculus (**DRC**).
- Differs from Relational Algebra by **being declarative**, not operational.
- **Safe expressions** are essential to ensure queries return finite, computable results.

Presented by
Om Gupta, Vikash Kumar, Aditya Raj,
Parkhi Jain, Adarsh Sharma & Gurnam Singh

THE END