Tutorial 1 — E-R Model, Basic Queries

Richard Wong
rk2wong@edu.uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

January 17, 2018

How to Best Make Use of Tutorials

Treat material presented in these slides as a supplement to Prof. Zarnett's lecture slides, given from a slightly different perspective.

If you notice that this perspective could be wrong, don't hesitate to speak up!

We will also walk through some exercises in tutorial to serve as practice.

The Entity-Relationship Model

In short, this model lets us represent:

- tables in a database with **mathematical sets**, and
- queries on the database with a language called the **relational algebra**.

Nouns of the Relational Model (1/2)

A **relation** R is equivalent to a table.

An **attribute** A is equivalent to a column of a table.

We describe the combination of relation and attributes with a **relation schema** of the form $R(A_1, A_2, \ldots, A_n)$.

$Student(student\_number, name, address)$ tells us:

- there is a table called `Student`, and
- it has attributes `student_number`, `name`, and `address`.

`name` is an attribute of the relation `Student`.

Nouns of the Relational Model (2/2)

The **contents** of a relation R are denoted $r(R)$.

$r(R)$ is an (unordered) set of (ordered) **n-tuples**.

Each **n-tuple** of $r(R)$ is equivalent to a row in the relation/table R.

The elements of each n-tuple correspond with the attributes $A_1, A_2, \ldots, A_n$ of R.

Nouns of the Entity-Relationship Model (1/5)

An **entity** [table] is equivalent to a relation, which is equivalent to a table.

- `Student` can be an entity in an E-R model.

However, we also use the term **entity** to describe an object in the real world, indistinguishable from other objects.

- A student with student number 20000000 and name Matt is an entity of type `Student`.

In this way, an entity [object] is equivalent to a row of an entity [table].

Nouns of the Entity-Relationship Model (2/5)

An **entity set** is a set of entities (in the object sense), and is equivalent to the contents of an entity table.

- The set of all students studying at the University of Waterloo might comprise the entity set `Student`.

For instance, `Student` = {(20000000, Matt), (20000001, Josh)}

Nouns of the Entity-Relationship Model (3/5)

Suppose we have another entity called `Course`, with attributes `course_id` and `title`.

Then the following are two possible entity sets:
`Student` = {(20000000, Matt), (20000001, Josh)}
`Course` = {(CS101, Intro to CS), (CS999, Super Hard CS)}

A **relationship** describes some connection between entities, in both the table sense and the object sense.

- In our E-R model, we can say `Student` takes `Course`.
- An individual student like Matt can take a course like Intro to CS.

Nouns of the Entity-Relationship Model (4/5)

A **relationship instance** describes how two entity instances are related. The following are each relationship instances:

- Matt takes Intro to CS: $takes_1$ = ((20000000, Matt), (CS101, Intro to CS))
- Josh also takes Intro to CS: $takes_2$ = ((20000001, Josh), (CS101, Intro to CS))
- Intro to CS is taught by Prof. X: $taughtby_1$ = ((CS101, Intro to CS), (1, Prof. X))

A **relationship set** is a set of relationship instances.

- In an E-R model, a relationship set represents how entities (in the table sense) are related.
  - `Students` take `Courses`: *Takes* = {$takes_1$, $takes_2$}
  - `Courses` are taught by `Instructors`: *TaughtBy* = {$taughtby_1$}

In an E-R model, we can use a table to represent a relationship set.
Such a table will have columns to identify which entities are related.

- The `StudentTakesCourse` table might have columns for `student_id` and `course_id`.
- This represents a relationship of **degree** 2, since it relates 2 entities.

Nouns of the Entity-Relationship Model (5/5)

An **attribute** is data used to describe an entity.

Attributes have a set of legal values called a **domain**, determined by the data type we ascribe to them in our relational schema.

Simple attributes contain a single value, e.g. `INT`, `VARCHAR(20)`, `ENUM`.

Composite attributes encapsulate other simple or composite attributes. Think someething similar to C `structs`.

- A composite attribute like an `Address` can be composed of the following attributes:
  - lot number,
  - street,
  - nullable apartment/unit number,
  - city,
  - province/state,
  - country, and
  - postal code.

A Note About `NULL`

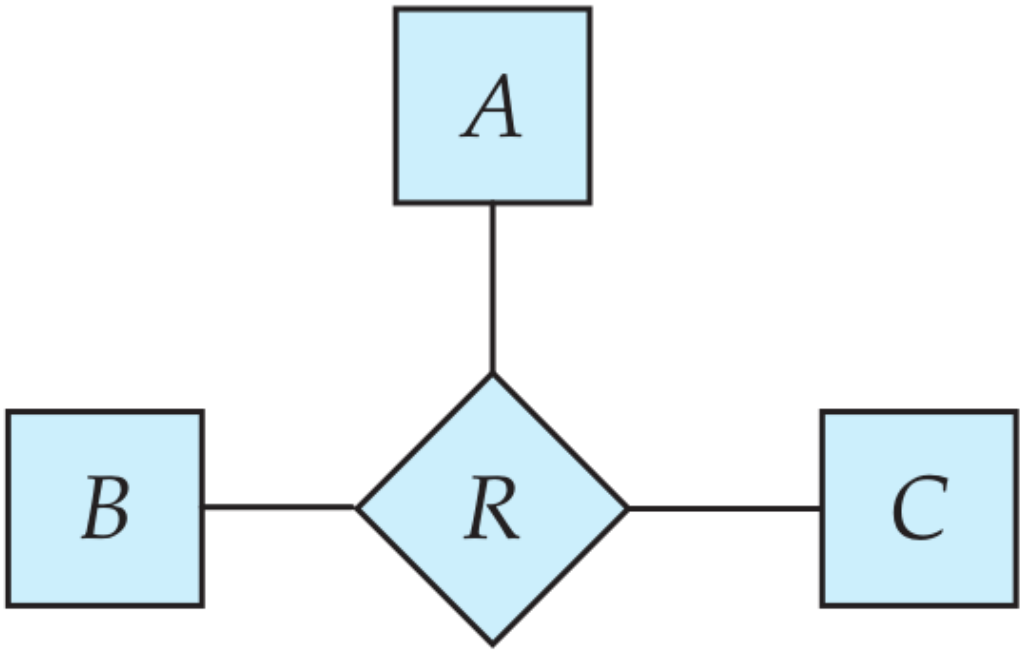
Use <column> `IS NULL` or <column> `IS NOT NULL` to perform null checks.

It typically does not make sense to have any other operators on `NULL` values.

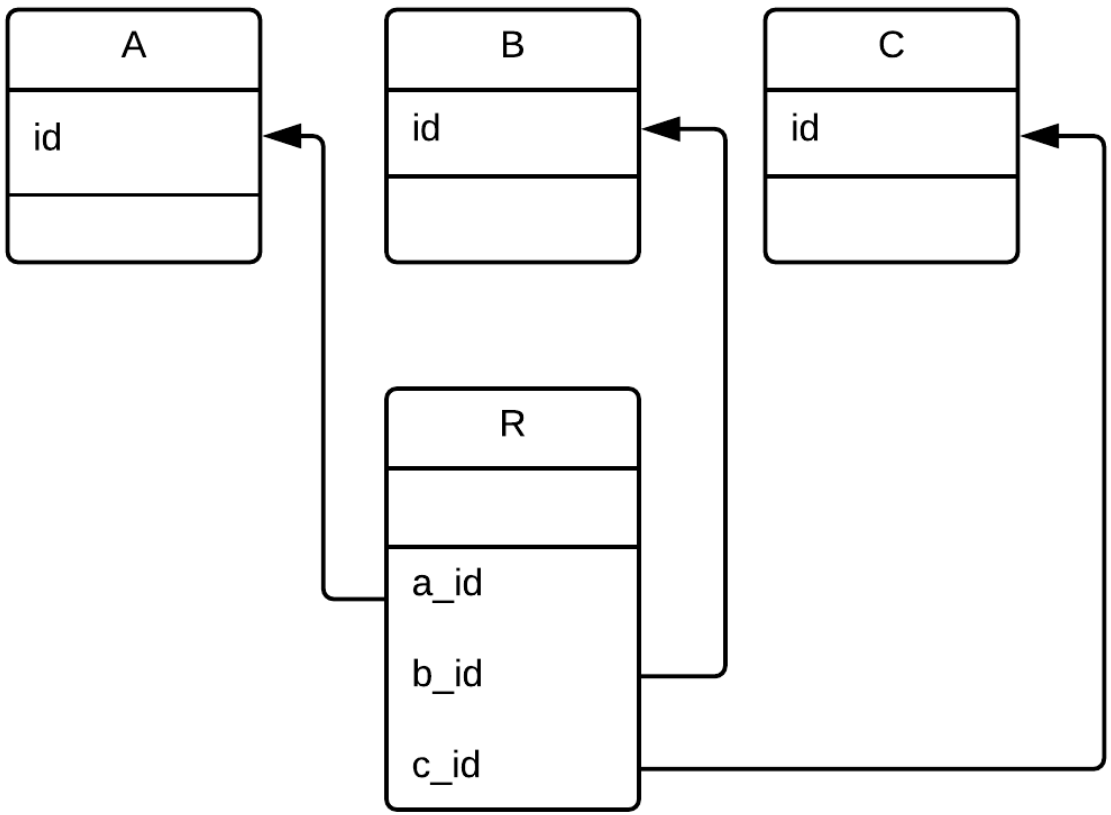
Treat `NULL` as "not having a value", as opposed to an "empty value" like 0 or "".

Let’s get some exercise! - E-R Modeling - 1.1

Translate the following E-R diagram into a set of database tables by writing (UML-esque) relational schemas:

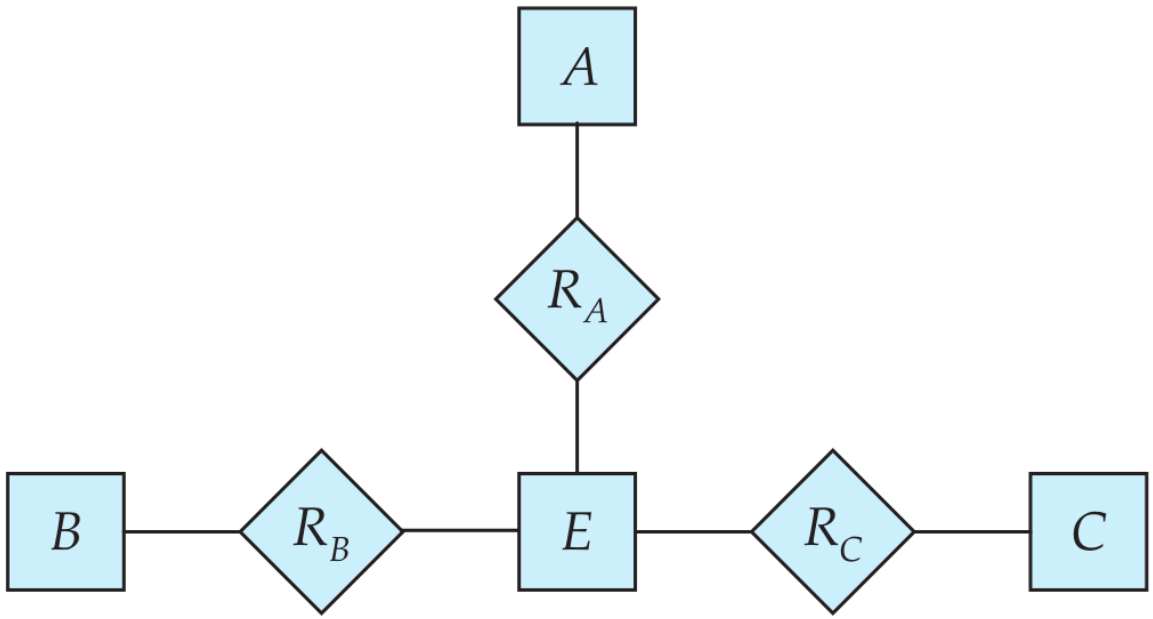


Let’s get some exercise! - E-R Modeling - 1.1

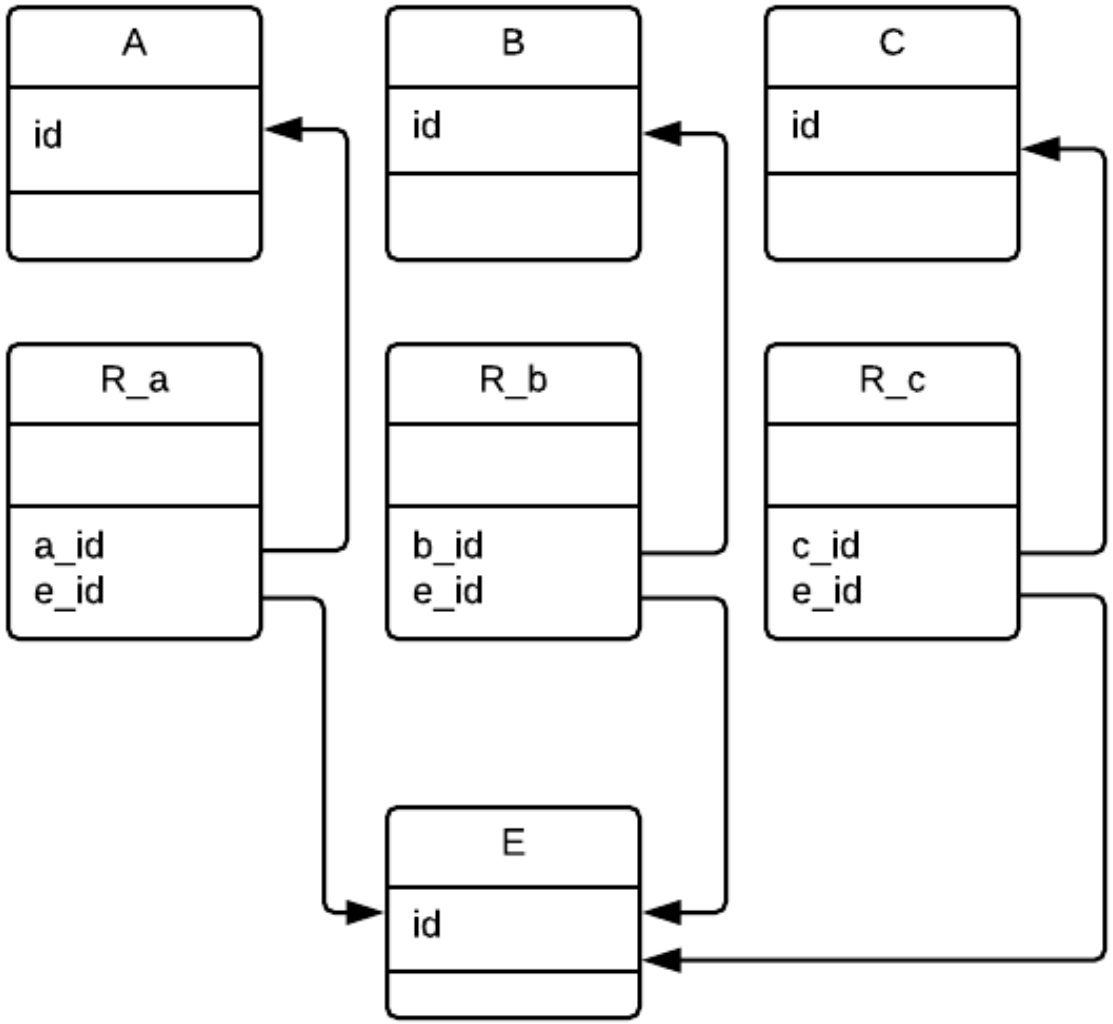


Let’s get some exercise! - E-R Modeling - 1.2

Translate the following E-R diagram into a set of database tables by writing (UML-esque) relational schemas:



Let’s get some exercise! - E-R Modeling - 1.2

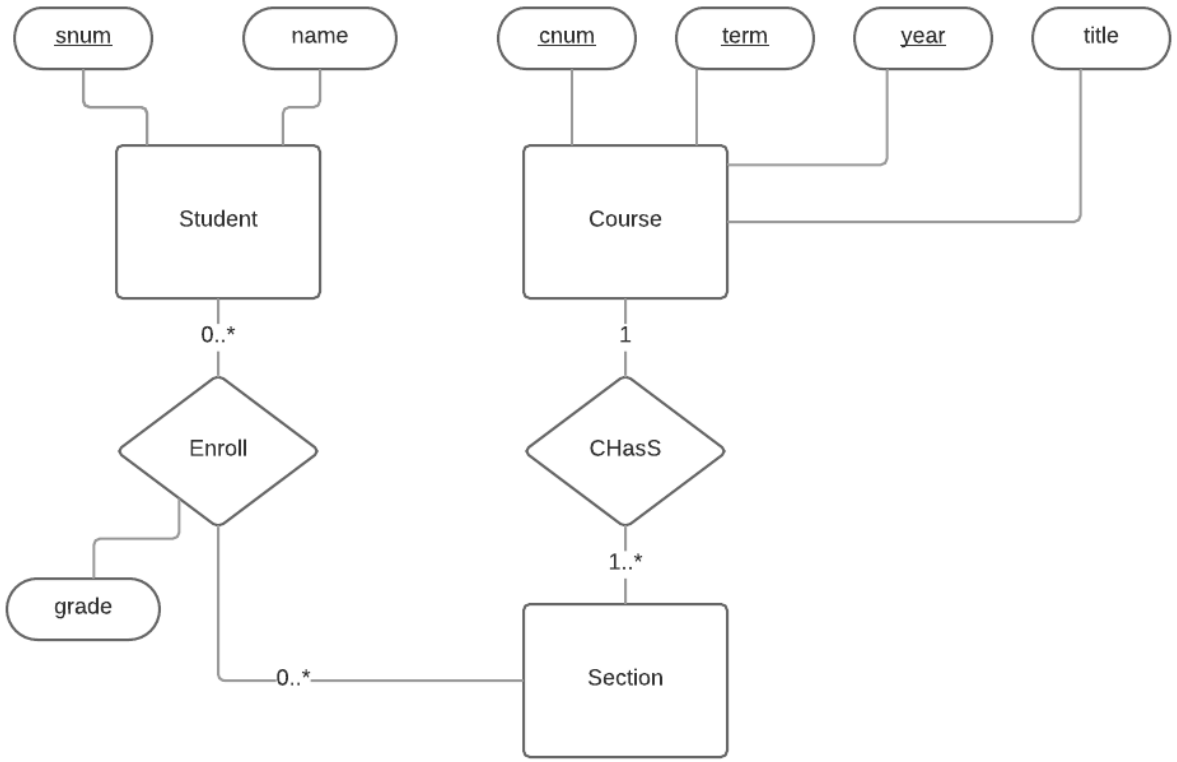


Let's get some exercise! - E-R Modeling - 1.3

Draw an E-R diagram to represent the following database specification for a simple Quest-like system:

- Students are identified by student numbers, and have names.
- Courses have course codes and titles.
- Courses are offered in a given term (W, S, F) in a given year.
- Courses are offered in one or more sections.
- Students enroll in sections of a course.
- Students get grades for the sections they are enrolled in at the end of a term.

Let’s get some exercise! - E-R Modeling - 1.3

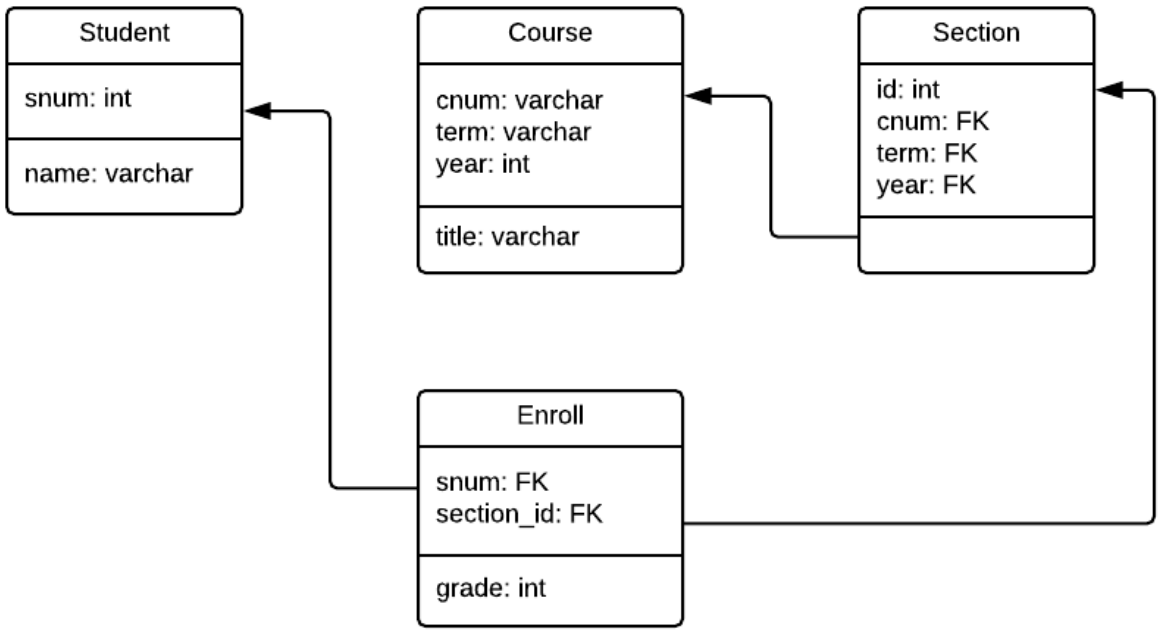


Let's get some exercise! - E-R Modeling - 1.4

Sketch out the tables that would implement your E-R diagram for the following specification:

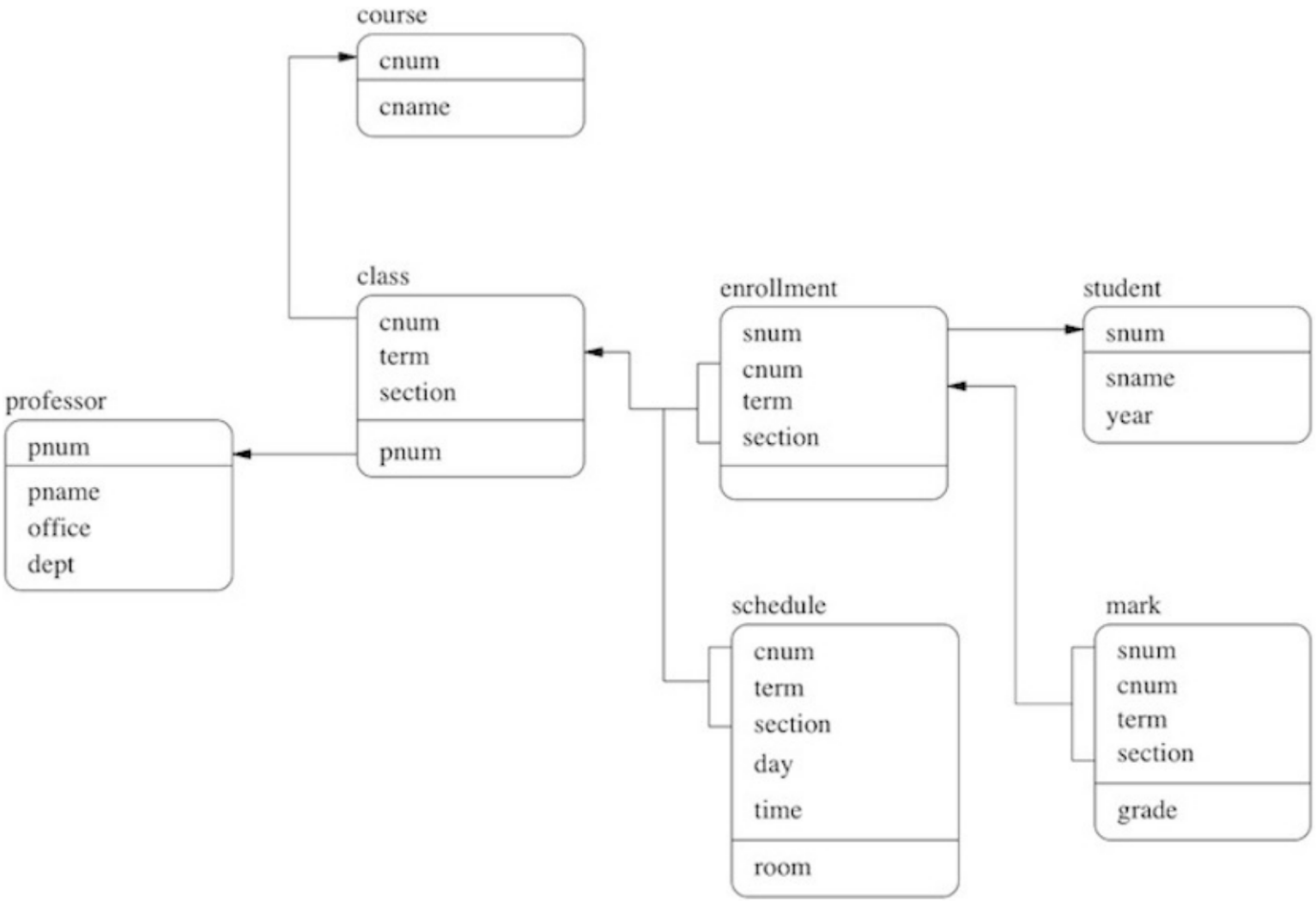
- Students are identified by student numbers, and have names.
- Courses have course codes and titles.
- Courses are offered in a given term (W, S, F) in a given year.
- Courses are offered in one or more sections.
- Students enroll in sections of a course.
- Students get grades for the sections they are enrolled in at the end of a term.

Let’s get some exercise! - E-R Modeling - 1.4



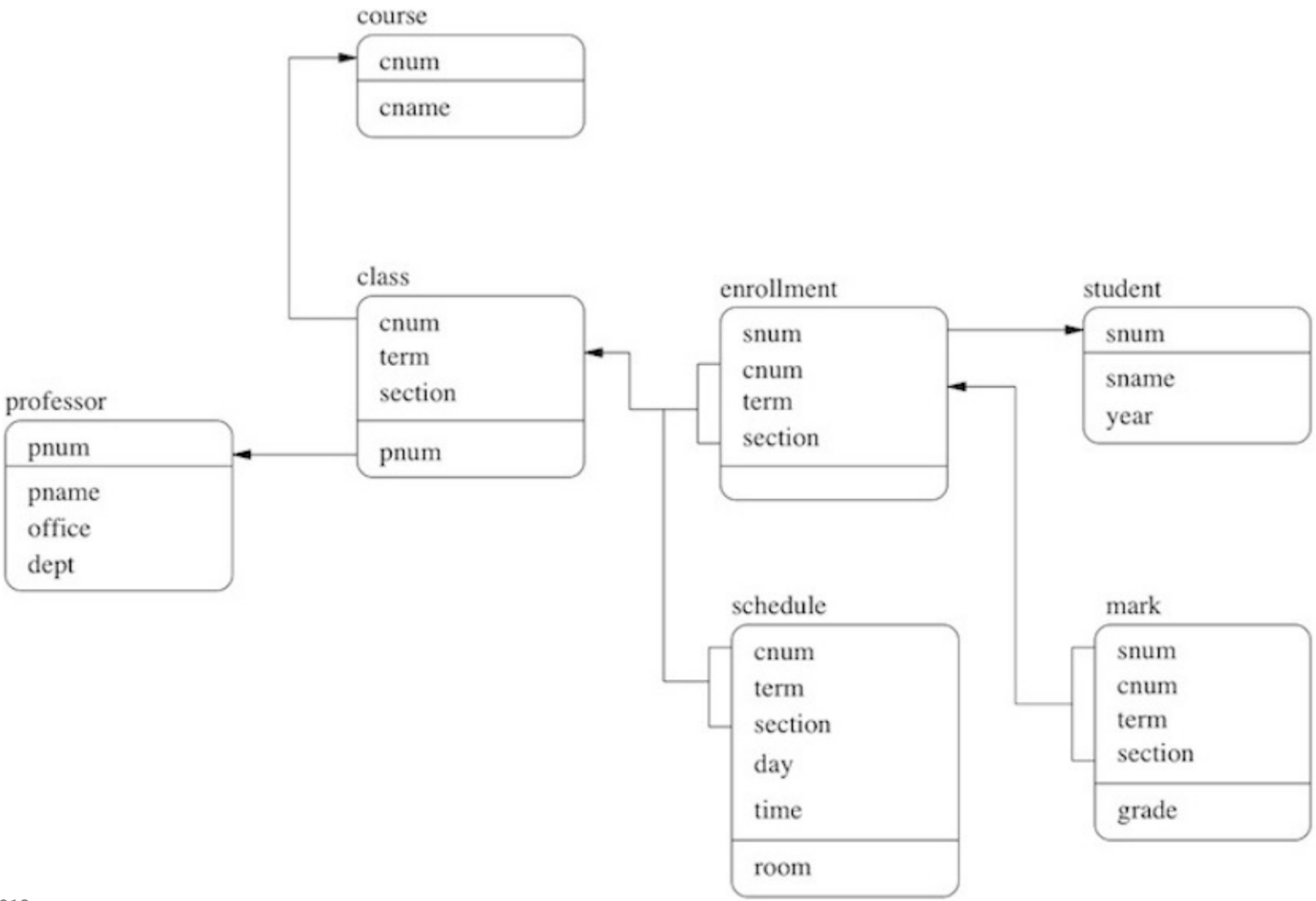
Let's get some exercise! - Relational Queries

Let's say we're given the following schema for a university database:



Let's get some exercise! - Relational Queries - 1.5

Find the names of all of the students that have not yet been given a grade in a class they have enrolled in.
Try writing the query in SQL, then in relational algebra.



Let's get some exercise! - Relational Queries - 1.5

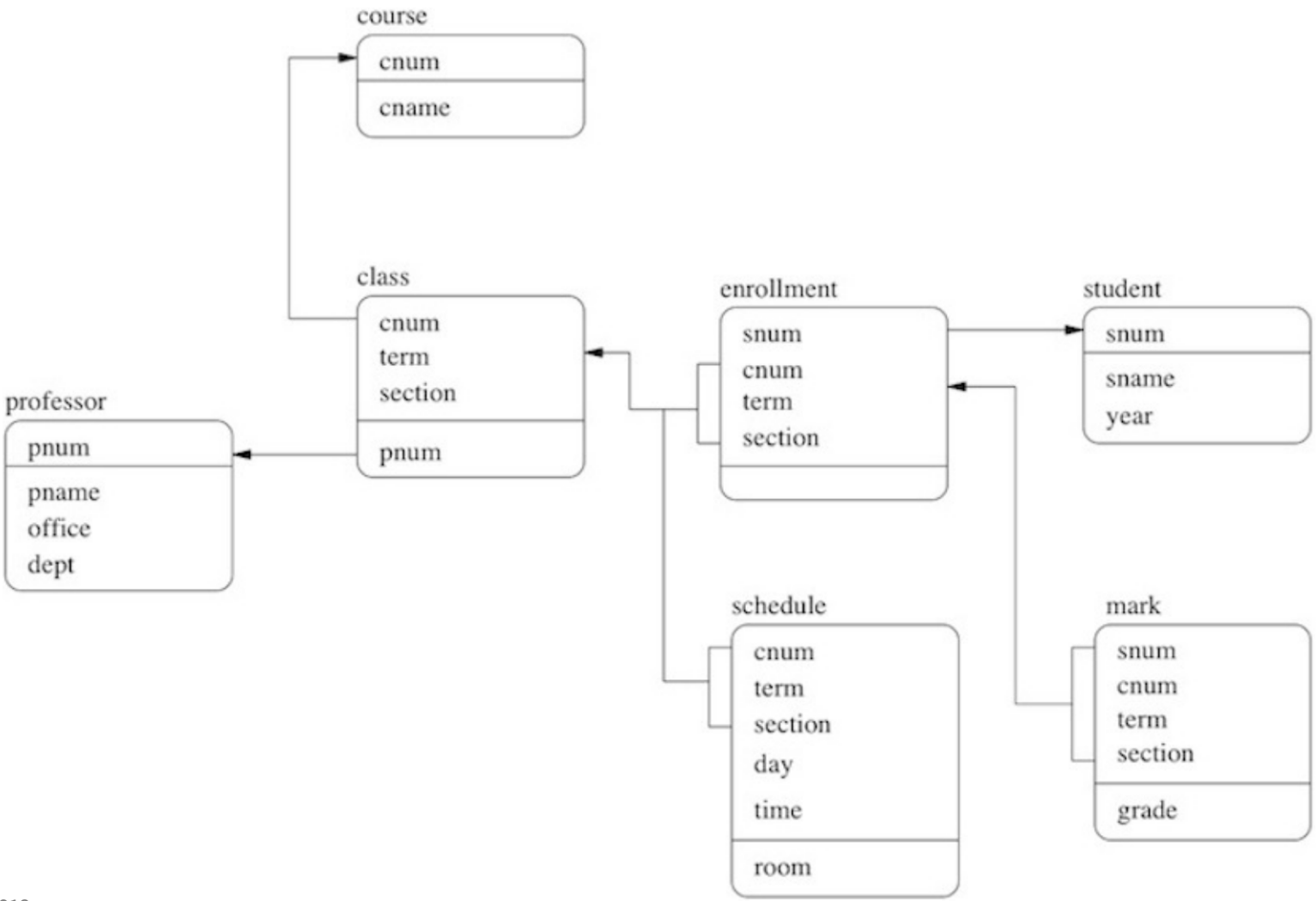
In the following solution, we assume that each enrollment necessarily has exactly one corresponding mark, and that a NULL grade represents a student not having a grade.

```
SELECT DISTINCT sname FROM student
JOIN mark
ON student.snum = mark.snum
WHERE grade IS NULL
```

$$\Pi_{sname}(\sigma_{isNull(grade)}(student \bowtie_{student.snum=mark.snum} mark))$$

Let's get some exercise! - Relational Queries - 1.6

Find all of the course codes along with the names of a student with the top grades in each course.
Try writing the query in SQL.

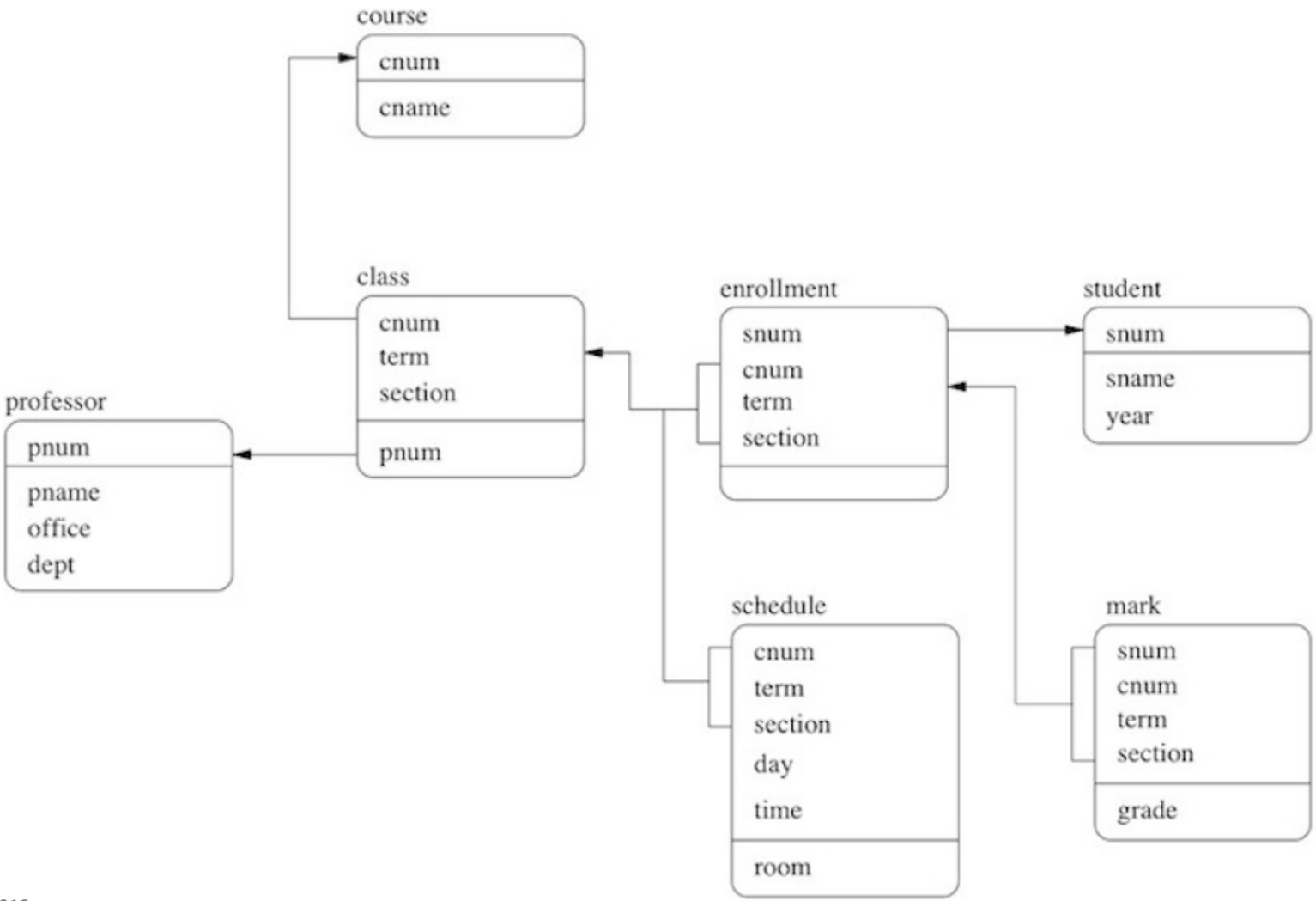


Let's get some exercise! - Relational Queries - 1.6

```
WITH BigJoin AS (
SELECT cnum, grade, snum, sname
FROM course
JOIN mark
ON course.cnum = mark.cnum
JOIN student
ON student.snum = mark.snum
GROUP BY cnum ),
TopGradePerCourse AS (
SELECT cnum, MAX(grade) AS topGrade
FROM BigJoin
GROUP BY cnum
)
SELECT cnum, sname
FROM BigJoin JOIN TopGradePerCourse
ON BigJoin.cnum = TopGradePerCourse.cnum
WHERE grade = topGrade
```

Let's get some exercise! - Relational Queries - 1.7

Find the names of all students and for each one, a count of the number of enrollments they have ever had.
Try writing the query in SQL.



Let's get some exercise! - Relational Queries - 1.7

```
WITH NonZeroCounts AS (
SELECT snum, COUNT(1) as count
FROM student
JOIN enrollment
ON student.snum = enrollment.snum
GROUP BY snum
)
SELECT sname, COALESCE[/ISNULL](count, 0)
FROM student
LEFT JOIN NonZeroCounts
ON student.snum = NonZeroCounts.snum
```

Fin

Questions or other topics I should talk about?