The Entity-Relationship Model

COMPSCI 2DB3: Databases

Jelle Hellings   Holly Koponen

Department of Computing and Software
McMaster University

Winter 2024

The relational data model

Data model

The rules by which real-world data can be represented and structured.

The relational data model

All data is modeled as a collection of *tables*.

Terminology: Schema and Instance

Schema Describe how stored data is structured.

- Conceptual schema: in terms of the *data model*.
- Physical schema: details in terms of *layout on disk*:
  files, file structure, auxiliary data structures (indices), ...

Instance The actual data (adhering to some schema).

Terminology: Schema and Instance

Schema: Describe how stored data is structured.

- Conceptual schema: in terms of the *data model*.
- Physical schema: details in terms of *layout on disk*: files, file structure, auxiliary data structures (indices), ….

Instance: The actual data (adhering to some schema).

Example: The relational data model

cgrades	($\underline{\text{id}}$,	name,	program,	age,	score)

The relational schema describing a table

Terminology: Schema and Instance

Schema Describe how stored data is structured.

- Conceptual schema: in terms of the *data model*.
- Physical schema: details in terms of *layout on disk*: files, file structure, auxiliary data structures (indices), ...

Instance The actual data (adhering to some schema).

Example: The relational data model

$$\underbrace{\underbrace{\textbf{cgrades}}_{\text{Table name}} \quad \underbrace{(\underline{\text{id}}, \quad \text{name}, \quad \text{program}, \quad \text{age}, \quad \text{score})}_{\text{Columns (\textit{attributes}) \& their domains}}}_{\text{The relational schema describing a table}}$$

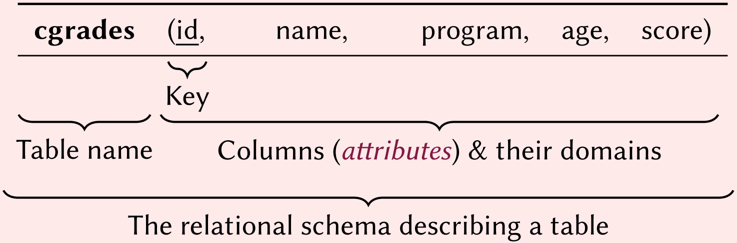
Terminology: Schema and Instance

Schema Describe how stored data is structured.

- Conceptual schema: in terms of the *data model*.
- Physical schema: details in terms of *layout on disk*: files, file structure, auxiliary data structures (indices), ...

Instance The actual data (adhering to some schema).

Example: The relational data model



cgrades (id, name, program, age, score)

Key

Table name

Columns (*attributes*) & their domains

The relational schema describing a table

Terminology: Schema and Instance

Schema Describe how stored data is structured.

- Conceptual schema: in terms of the *data model*.
- Physical schema: details in terms of *layout on disk*: files, file structure, auxiliary data structures (indices), ...

Instance The actual data (adhering to some schema).

Example: The relational data model

cgrades	(<u>id</u>,	name,	program,	age,	score)	
	53666	Jones	cs	18	3.4	Instance
	53688	Smith	ee	18	3.2	
	53650	Smith	math	19	3.8	
	53831	Madayan	music	11	1.8	
	53832	Guldu	music	12	2.0	

How to create a schema?

Requirements Analysis

- What data can this application collect?
- What data does this application need?
- What does the application do with the data?

How to create a schema?

Requirements Analysis

- What data can this application collect?
- What data does this application need?
- What does the application do with the data?

Is a database management system the right fit?
E.g., big data analytics, machine learning, visualization, …

How to create a schema?

Requirements Analysis

- What data can this application collect?
- What data does this application need?
- What does the application do with the data?

Is a database management system the right fit?
E.g., big data analytics, machine learning, visualization, . . .

What about agile development?

It is complex, costly, and sometimes impossible to refactor away *wrong data choices*. E.g.,

- How do you add missing data after the fact?
- How do you restructure data to improve performance?

Agile development *requires* expertise to make the right choices at the right time.

Requirements Analysis: the entity-relationship data model

Relational data: Tables represent everything

Faculty(fid, name, location, mail),
Course(cid, name, year, credits).
Teaches(fid, cid).

Requirements Analysis: the entity-relationship data model

Relational data: Tables represent everything

$$\left.\begin{array}{l} \textit{Faculty}(\underline{\text{fid}}, \text{name, location, mail}), \\ \textit{Course}(\underline{\text{cid}}, \text{name, year, credits}). \end{array}\right\} \textit{Entities}$$

$$\left.\begin{array}{l} \textit{Teaches}(\underline{\text{fid}}, \underline{\text{cid}}). \end{array}\right\} \textit{Relationship}$$

Requirements Analysis: the entity-relationship data model

Relational data: Tables represent everything

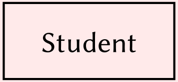
$$\left.\begin{array}{l}\textit{Faculty}(\underline{\text{fid}}, \text{name, location, mail}),\\ \textit{Course}(\underline{\text{cid}}, \text{name, year, credits}).\end{array}\right\}\textit{Entities}$$

$$\left.\textit{Teaches}(\underline{\text{fid}}, \underline{\text{cid}}).\right\}\textit{Relationship}$$

Entity-Relationship data model

- A *semantic* data model: Specify *meaning* and not *representation*.
- Can model high-level *concepts*: entities, relationships, attributes, ...
- Can easily be translated to well-designed tables (*representation*).
- Resulting ER-diagrams are easier to document and discuss with *non-specialists*.

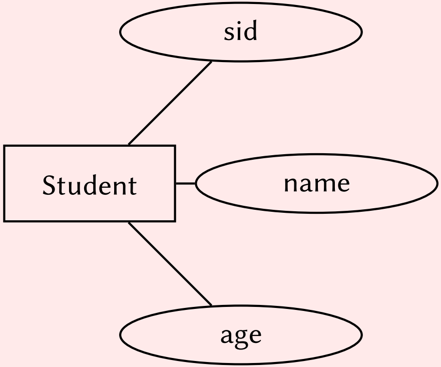
A step-by-step example: Students enrolled in courses



Student

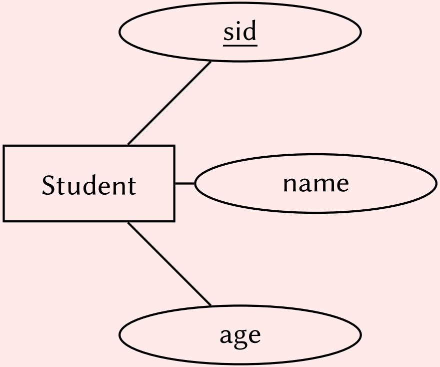
The *entity set* student.

A step-by-step example: Students enrolled in courses



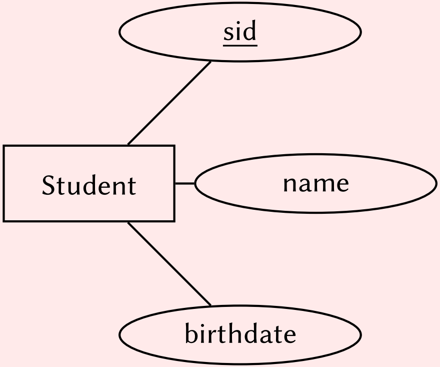
Students have *attributes*: student id, name, and age.

A step-by-step example: Students enrolled in courses



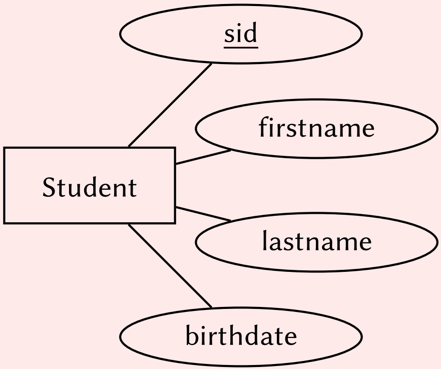
Student id is a *key*: a unique identifier.

A step-by-step example: Students enrolled in courses



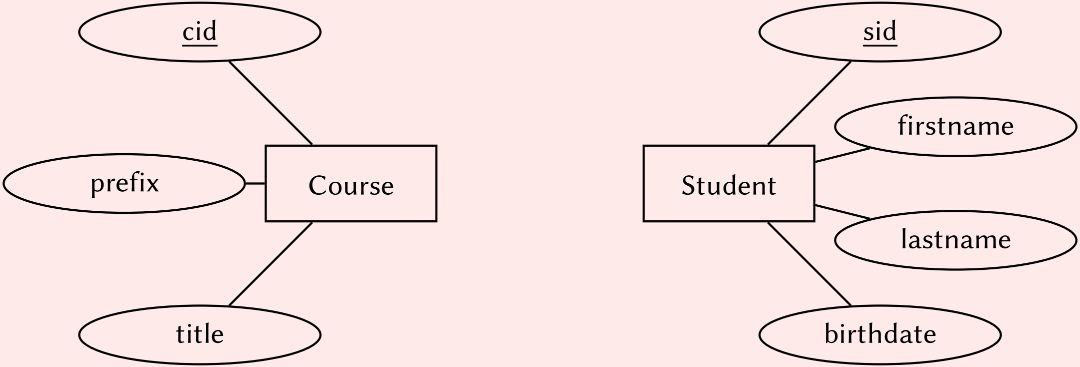
Age is a *derived* attribute: birthdate is easier to maintain!

A step-by-step example: Students enrolled in courses



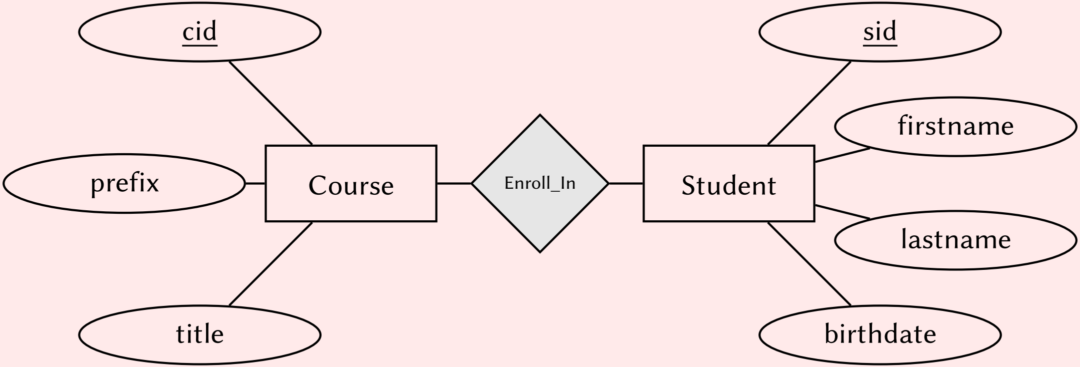
Names are *complex*: does the application need first and last names?

A step-by-step example: Students enrolled in courses



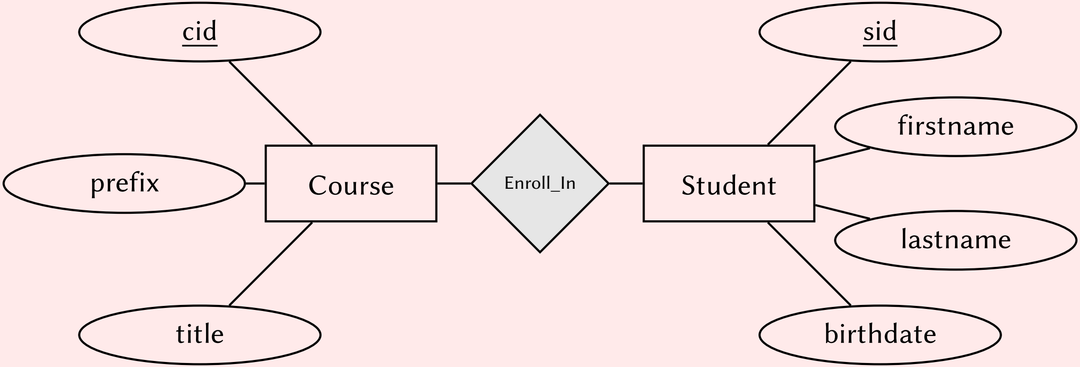
Another *entity*: a course.

A step-by-step example: Students enrolled in courses



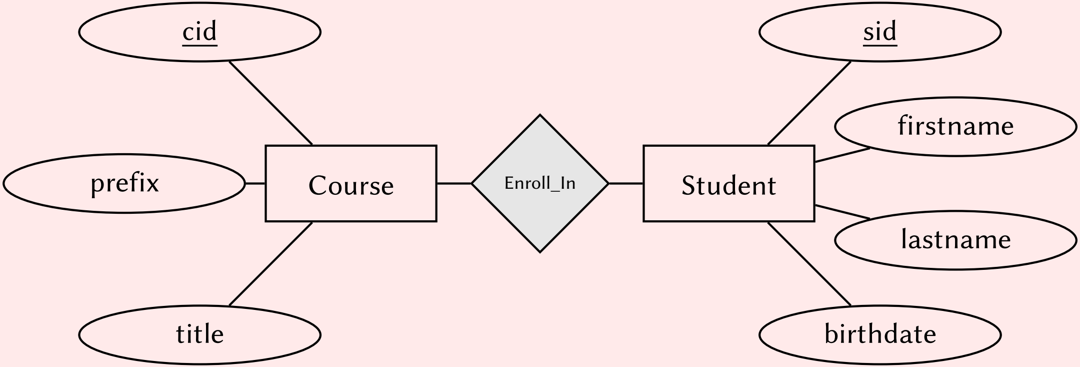
Each student can enroll in classes: a *relationship set*.

A step-by-step example: Students enrolled in courses



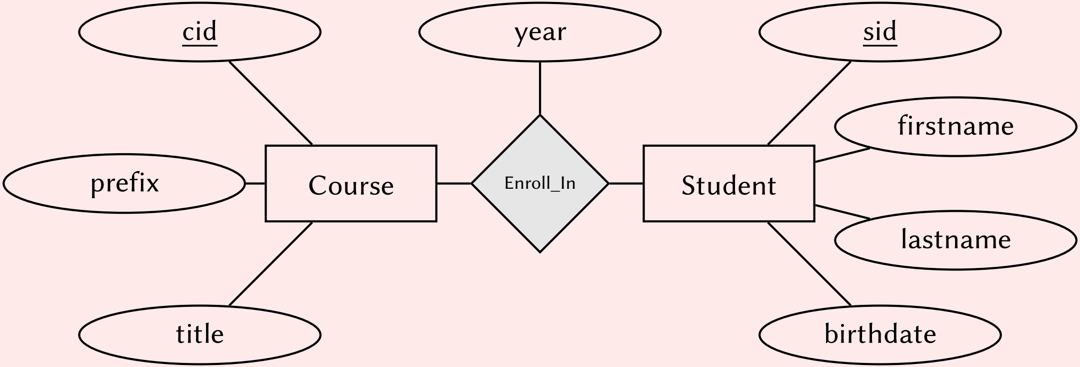
This is a *many-to-many relationship set*.

A step-by-step example: Students enrolled in courses



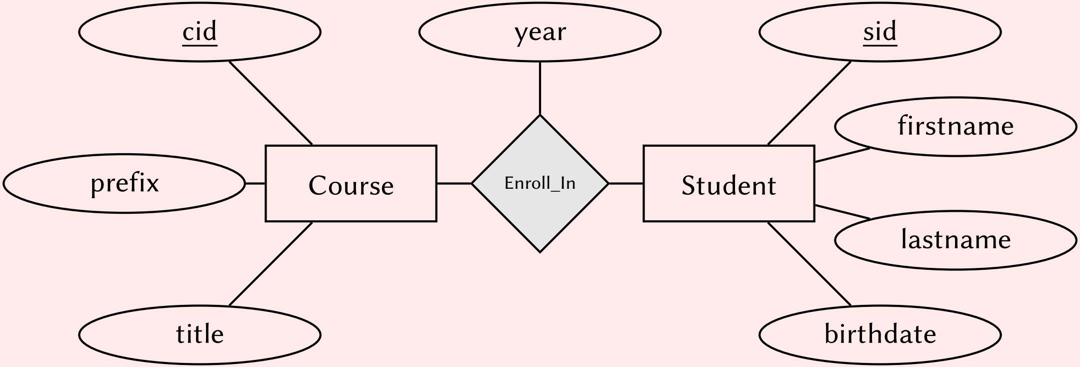
Relationships are *unique*: A student cannot take a single course multiple times.

A step-by-step example: Students enrolled in courses



Relationship sets can have attributes: the *year* in which the course was taken.

A step-by-step example: Students enrolled in courses



Even with relationship attributes: Students can take a course only once!

Intermission: Names…

Question: Did you ever have problems with your name?

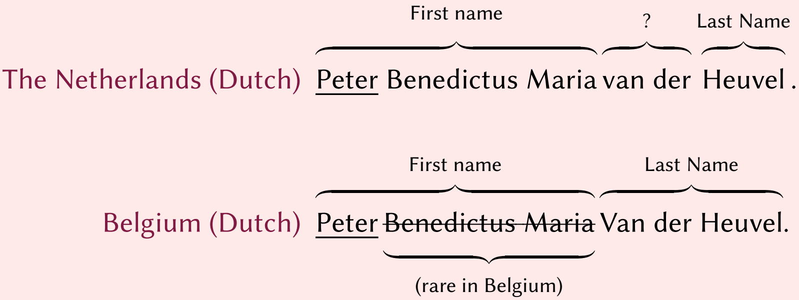
Vote at `https://strawpoll.com/cp6fysdxr`.
Or: go to `https://strawpoll.live` and use the code **112128**.

Intermission: Names—an example

$$\text{The Netherlands (Dutch)}\quad \overbrace{\underline{\text{Peter}}\ \text{Benedictus Maria}}^{\text{First name}}\ \overbrace{\text{van der}}^{?}\ \overbrace{\text{Heuvel}}^{\text{Last Name}}.$$

Some useful background reading: Falsehoods Programmers Believe About Names.

Intermission: Names—an example



First name

?

Last Name

The Netherlands (Dutch) Peter Benedictus Maria van der Heuvel .

First name

Last Name

Belgium (Dutch) Peter ~~Benedictus Maria~~ Van der Heuvel.

(rare in Belgium)

Some useful background reading: Falsehoods Programmers Believe About Names.

Intermission: Names—an example

The Netherlands (Dutch)

$$\overbrace{\underline{\text{Peter}}\ \text{Benedictus Maria}}^{\text{First name}}\ \overbrace{\text{van der}}^{?}\ \overbrace{\text{Heuvel}}^{\text{Last Name}}\ .$$

Belgium (Dutch)

$$\overbrace{\underline{\text{Peter}}\ \underbrace{\sout{\text{Benedictus Maria}}}_{\text{(rare in Belgium)}}}^{\text{First name}}\ \overbrace{\text{Van der Heuvel}}^{\text{Last Name}}.$$

USA (English)

$$\overbrace{\text{Peter}}^{\text{First name}}\quad \underbrace{\text{‘}\underline{\text{Pete}}\text{’}}_{\text{Nickname}}\ \overbrace{\text{Benedictus Maria}}^{\text{Middle names}}\ \overbrace{\underbrace{\text{Vanderheuvel}}_{\text{‘Vanderhill’?}}}^{\text{Last Name}}.$$

Some useful background reading: Falsehoods Programmers Believe About Names.

Intermission: What to do with names?

Ask yourself: Why do you need the name?

1. Do not put up limitations, unless you have a specific goal.
2. Be clear to users so they can provide correct info. E.g.,

   Informal Ask *what they want* to be called (e.g., Hi!).

   Formal Ask *what they want* to be called formally (e.g., letters).

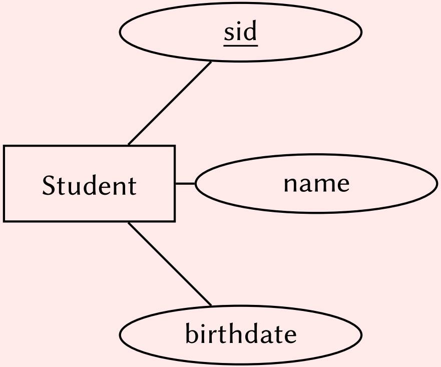
   Billing Ask the name *as used by their bank*.

   Shipping Ask the name *as used by the mail carrier*.

   Travel Ask the name *as printed in their travel documents*.
3. Have *sensible defaults*: e.g., prefill billing names with account information.

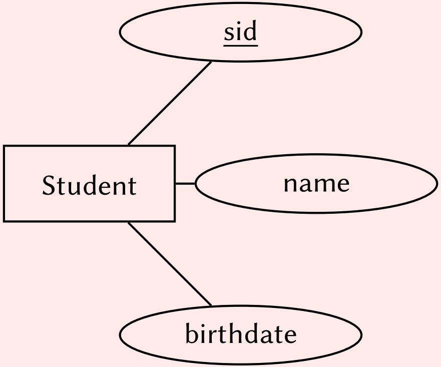
There is no standard solution that works in all cases!

The entity-relationship model: Entities



- An *entity* is an object in the real world.
- Entities are described via a set of *attributes*.
- Attributes have a *domain* (e.g., text, number, date).

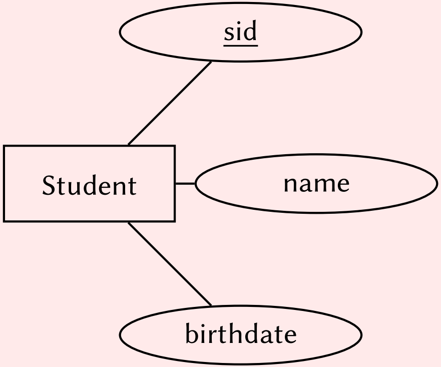
The entity-relationship model: Entities



- An *entity* is an object in the real world.
- Entities are described via a set of *attributes*.
- Attributes have a *domain* (e.g., text, number, date).

- *Key*: set of attributes that identify an entity.
- All keys are *candidate keys*.
- One is chosen as the *primary key*.

The entity-relationship model: Entities

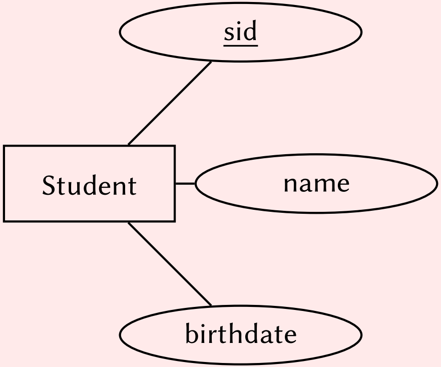


- An *entity* is an object in the real world.
- Entities are described via a set of *attributes*.
- Attributes have a *domain* (e.g., text, number, date).

- *Key*: set of attributes that identify an entity.
- All keys are *candidate keys*.
- One is chosen as the *primary key*.

- An *entity set*: collection of similar entities.
- An *instance*: snapshot of the entities (“data”).

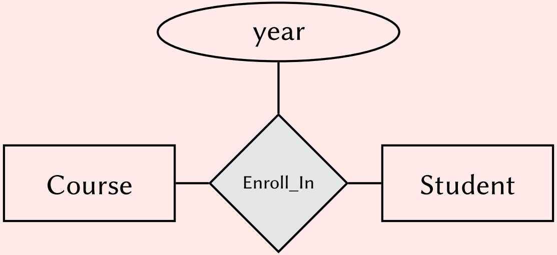
The entity-relationship model: Entities



Instance

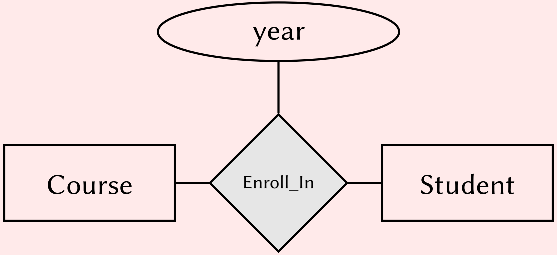
sid	name	birthdate
53666	Jones	May 18, 1999
53688	Smith	June 2, 1994
53650	Smith	December 21, 1995
53831	Madayan	February 7, 2000
53832	Guldu	April 1, 1991

The entity-relationship model: Relationships



- A *relationship* relates two or more entities.
- Standard relationship: *many-to-many*:
  Students can enroll in *many* courses, courses can have *many* enrolled students.
- Relationships can have additional *attributes*.

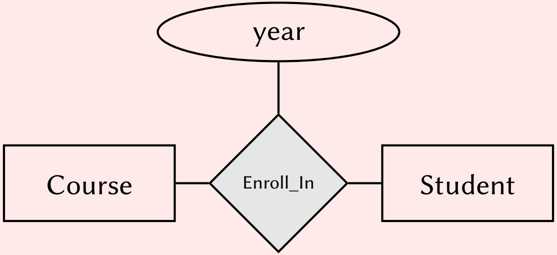
The entity-relationship model: Relationships



- A *relationship* relates two or more entities.
- Standard relationship: *many-to-many*:
  Students can enroll in *many* courses, courses can have *many* enrolled students.
- Relationships can have additional *attributes*.

- A *relationship set*: collection of similar relationships between entities.
- A *instance*: snapshot of the relations (“data”).

The entity-relationship model: Relationships



Instance

Course	*Student*	year
Course_1	*Student_1*	2019
Course_1	*Student_2*	2020

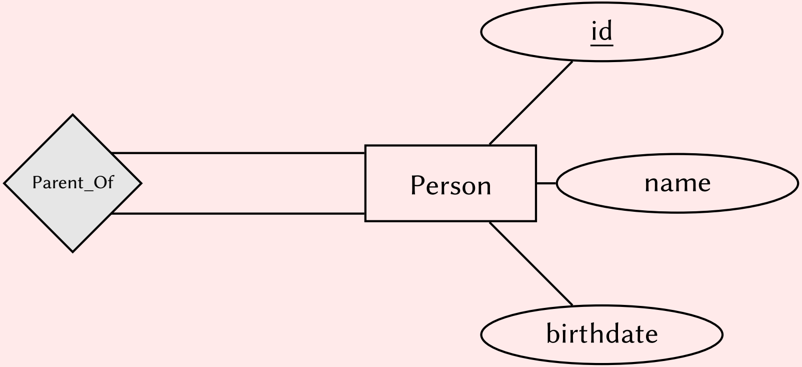
Course	*Student*	year
Course_2	*Student_1*	2019
Course_2	*Student_3*	2021

More on relationships

Relationships are at the core of the entity-relationship model.

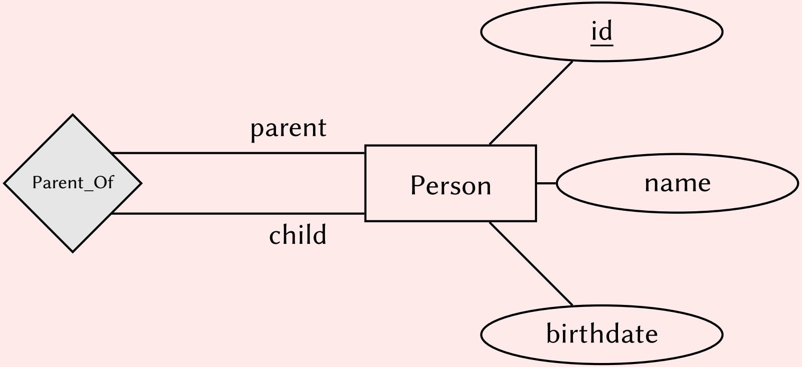
- Self-referential relationship sets.
- *n*-ary relationship sets.
- Key constraints.

Self-relationships



- Entities in an entity set can be related to each others.
  E.g., *Persons* can be *parents*, *friends*, *partners*, *colleagues*.

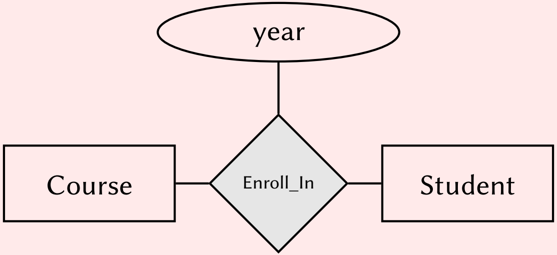
Self-relationships



- Entities in an entity set can be related to each others.
  E.g., *Persons* can be *parents*, *friends*, *partners*, *colleagues*.
- Typically, one names the *roles* in such relationships sets.

Ternary relationships

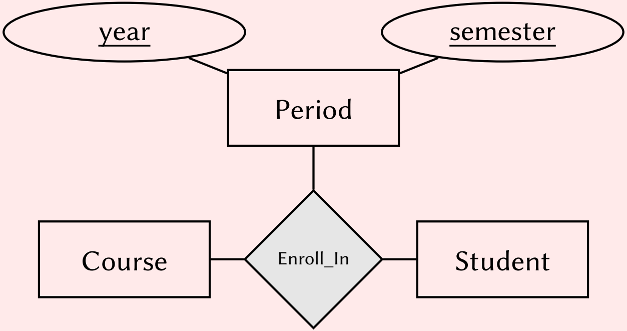
What if a student can enroll several times?



Pairs (*student*, *course*) must be unique.

Ternary relationships

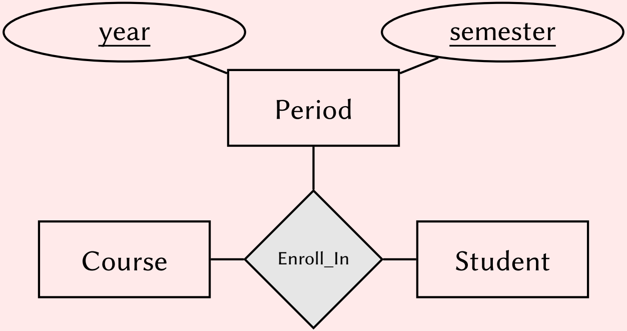
What if a student can enroll several times?



Pairs (*student*, *course*, *period*) must be unique.

Ternary relationships

What if a student can enroll several times?



Ternary relationships are rare: this can be modeled in many other ways.

Types of binary relations (2DM3)

A binary relation $R \subseteq A \times B$ is

Many-to-Many if there are no restrictions on the relation.

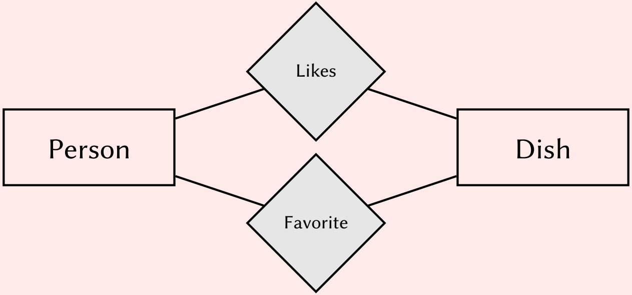
One-to-Many if each $a \in A$ is related to *at-most-one* $b \in B$.

- If A has *total participation*: R is a *function* of A,
  each $a \in A$ is related to *exactly-one* $b \in B$.

One-to-One each A is related to *at-most-one* B and vice-versa.

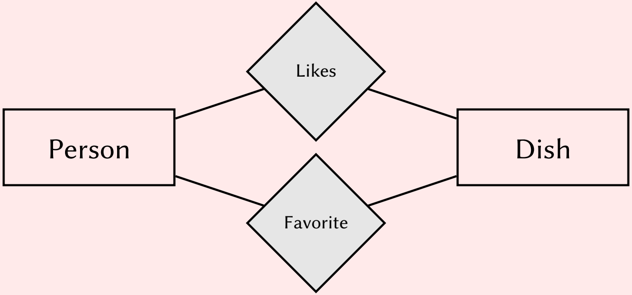
- If A has *total participation*: R is an *injection*,
  each $a \in A$ is related to exactly-one-and-unique $b \in B$.
- If B has *total participation*: R is an *surjection*,
  each $b \in B$ is related to exactly-one-and-unique $a \in A$.
- If A, B have *total participation*: R is a *bijection*,
  there is a one-to-one mapping between all $a \in A$s and $b \in B$s.

Constraints on relationships



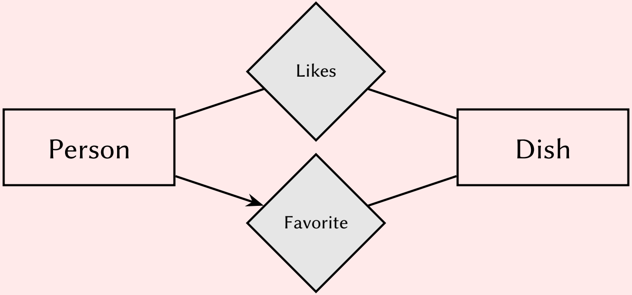
A person *likes* many dishes, but has *one* favorite dish.

Constraints on relationships



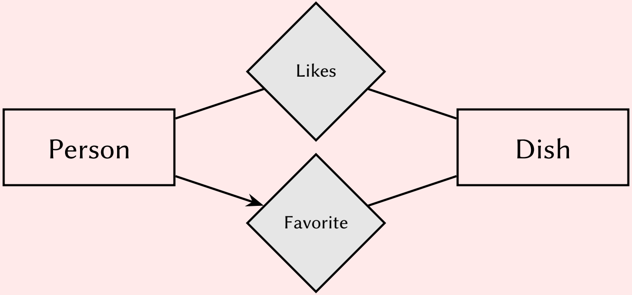
This says: a person has *many* favorite dishes!

Constraints on relationships



Key constraint: A person has *at-most-one* favorite dish.

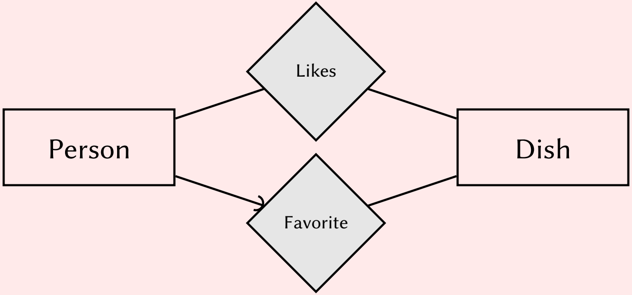
Constraints on relationships



Key constraint: A person has *at-most-one* favorite dish.

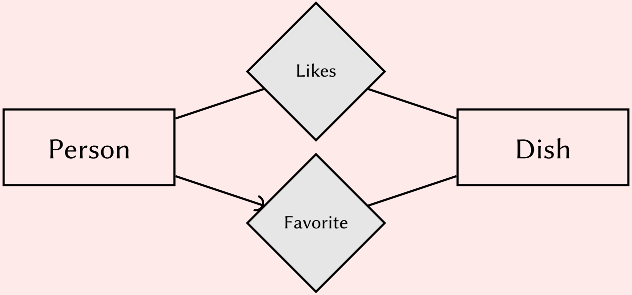
$E \longrightarrow R$: entity E partakes at-most-once in a R-relationship (partial).

Constraints on relationships



Key constraint: A person has *exactly-one* favorite dish.

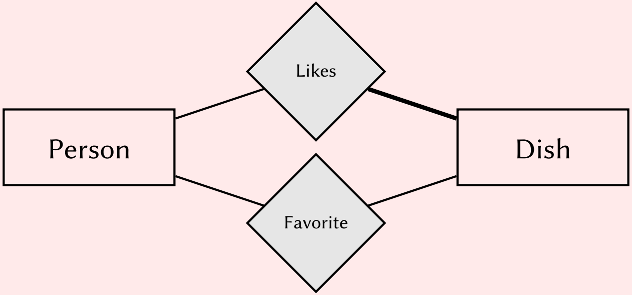
Constraints on relationships



Key constraint: A person has *exactly-one* favorite dish.

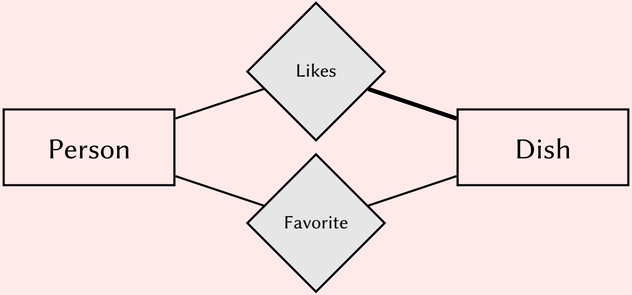
$E \longrightarrow R$: entity E partakes exactly-once in a R-relationship (total).

Constraints on relationships



Participation constraint: Every dish is *liked*.

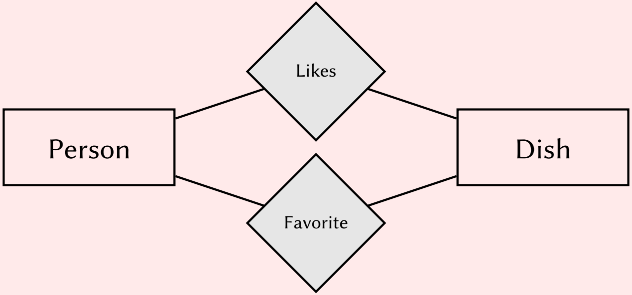
Constraints on relationships



Participation constraint: Every dish is *liked*.

E ━━━ R: entity E partakes at-least-once in a R-relationship (total).

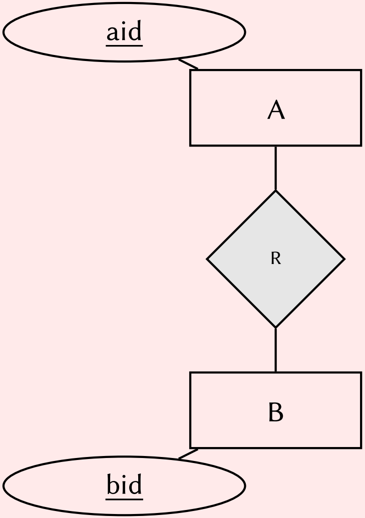
Constraints on relationships



Remarks on notation

- We follow the notation of the textbook.
- Many other sources use different notations (e.g., arrows reversed).
- We use dedicated notation for exactly-once participation ($\longrightarrow\!\!\!\ni$).

Key constraints



Entities A
aid
α
β
γ
δ

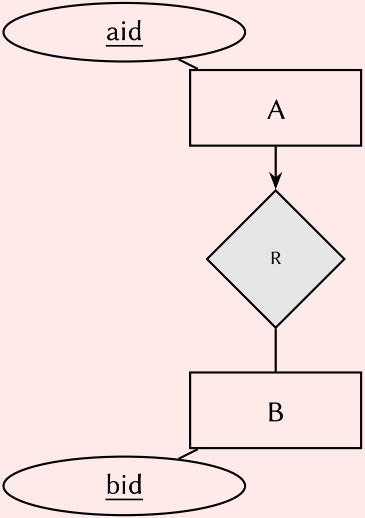
Relationships R

aid	**bid**
α	1
α	2
β	1
β	2
γ	3

Entities B
bid
1
2
3
4

many-to-many

Key constraints



Entities A

aid
α
β
γ
δ

Relationships R

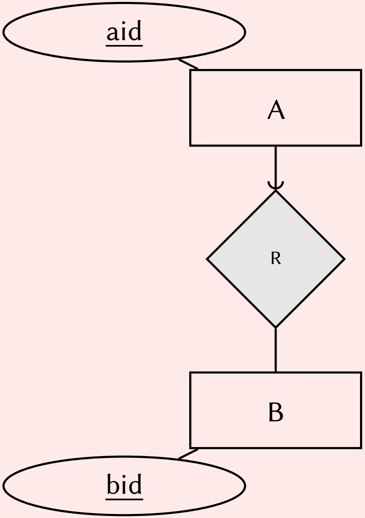
aid	bid
α	1
~~α~~	~~2~~
β	1
~~β~~	~~2~~
γ	3

one-to-many
(partial)

Entities B

bid
1
2
3
4

Key constraints



Entities A

aid
α
β
γ
~~δ~~

Relationships R

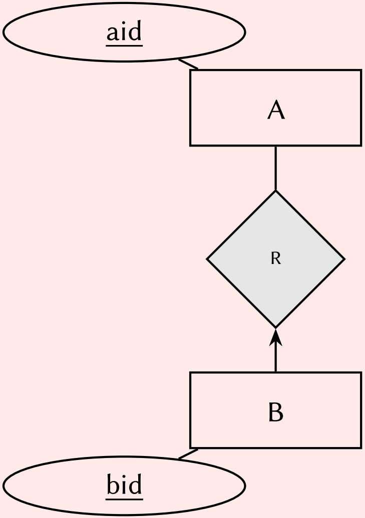
aid	bid
α	1
~~α~~	~~2~~
β	1
~~β~~	~~2~~
γ	3

Entities B

bid
1
2
3
4

one-to-many
(total)
(function)

Key constraints



Entities A

aid
α
β
γ
δ

Relationships R

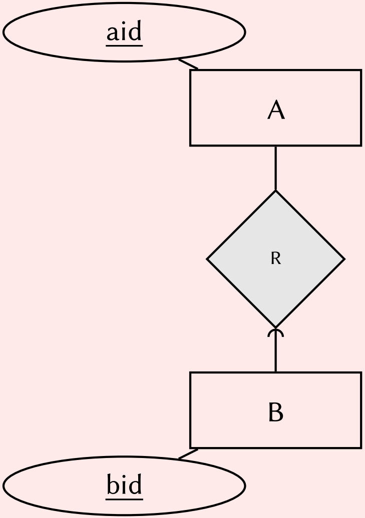
aid	bid
α	1
α	2
~~β~~	~~1~~
~~β~~	~~2~~
γ	3

Entities B

bid
1
2
3
4

many-to-one
(partial)

Key constraints



Entities A

aid
α
β
γ
δ

Relationships R

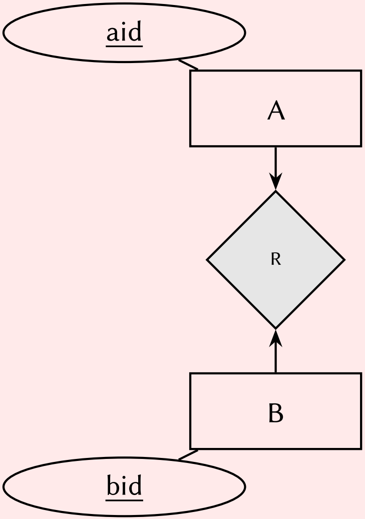
aid	bid
α	1
α	2
~~β~~	~~1~~
~~β~~	~~2~~
γ	3

Entities B

bid
1
2
3
~~4~~

many-to-one
(total)

Key constraints



Entities A

aid
α
β
γ
δ

Relationships R

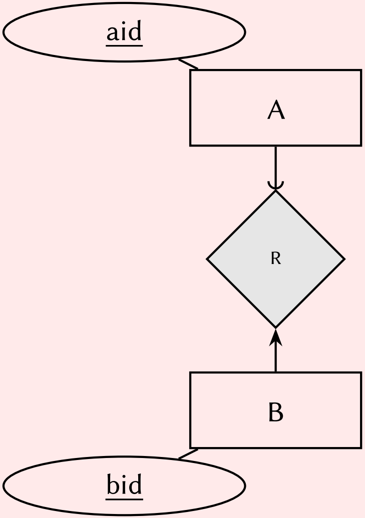
aid	**bid**
α	1
~~α~~	~~2~~
~~β~~	~~1~~
β	2
γ	3

Entities B

bid
1
2
3
4

one-to-one
(partial, partial)

Key constraints



Entities A

aid
α
β
γ
~~δ~~

Relationships R

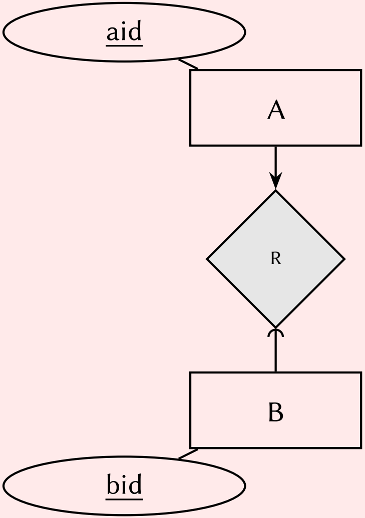
aid	bid
α	1
~~α~~	~~2~~
~~β~~	~~1~~
β	2
γ	3

Entities B

bid
1
2
3
4

one-to-one
(total, partial)
(injection)

Key constraints



Entities A

aid
α
β
γ
δ

Relationships R

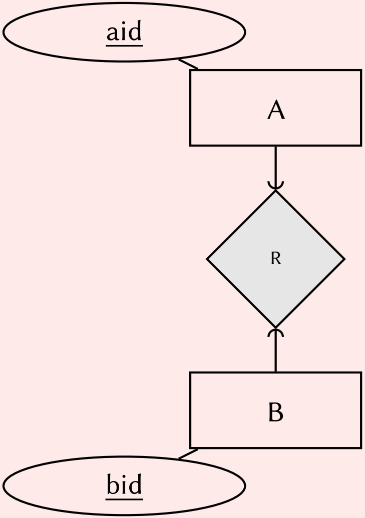
aid	**bid**
α	1
~~α~~	~~2~~
~~β~~	~~1~~
β	2
γ	3

Entities B

bid
1
2
3
~~4~~

one-to-one
(partial, total)
(surjection)

Key constraints



Entities A

aid
α
β
γ
~~δ~~

Relationships R

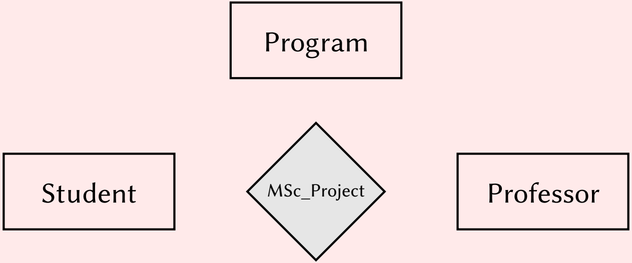
aid	bid
α	1
~~α~~	~~2~~
~~β~~	~~1~~
β	2
γ	3

Entities B

bid
1
2
3
~~4~~

one-to-one
(total, total)
(bijection)

Key constraints and ternary relationships–1

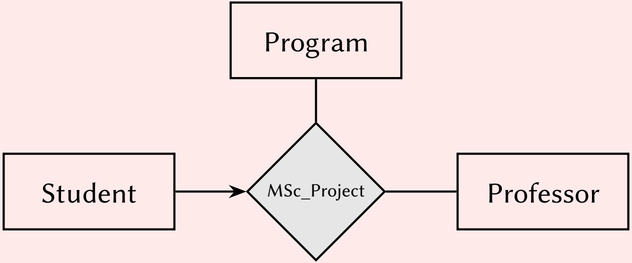


- A student can do *one* Master Project.
- Each project has a supervising professor and is performed in a degree program.

Question: Where should the key constraints go?

Vote at `https://strawpoll.com/3f6eovfz8`.
Or: go to `https://strawpoll.live` and use the code **265451**.

Key constraints and ternary relationships–1

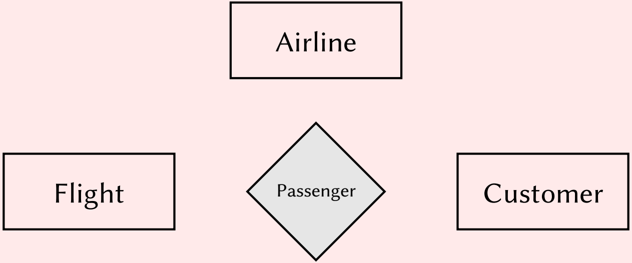


- A student can do *one* Master Project.
- Each project has a supervising professor and is performed in a degree program.

Answer: Constraint on Student

- Key constraint for the participation of students.
- Professors can supervise *many* Master Projects.
- *Many* students can perform a Master Project within a program.

Key constraints and ternary relationships–2

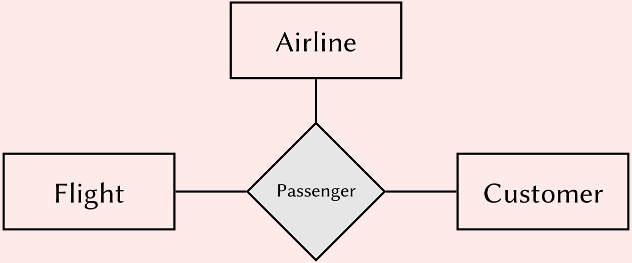


Each passenger (customer) of a flight bought a single ticket via a single airline.

Question: Where should the key constraints go?

Vote at `https://strawpoll.com/gwbwsh9hy`.
Or: go to `https://strawpoll.live` and use the code **241367**.

Key constraints and ternary relationships–2



Each passenger (customer) of a flight bought a single ticket via a single airline.

Answer: This is not a key constraint

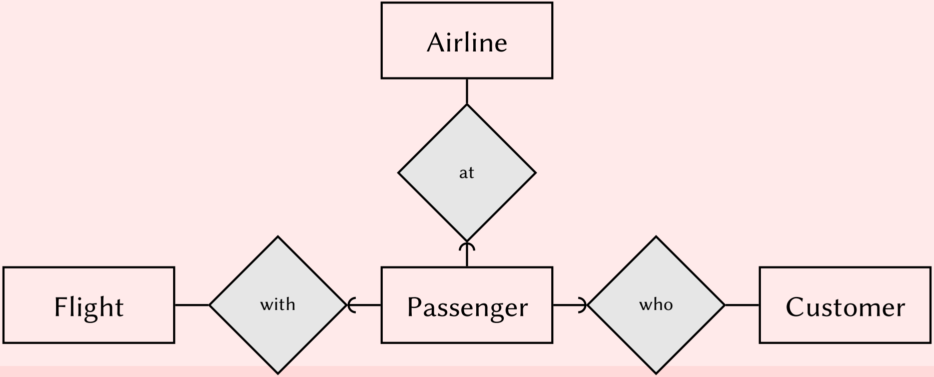
- Customers can be on *many* flights.
- Airlines can sell tickets for *many* flights.
- Flights can have *many* customers, each from different airlines.

Key constraints and ternary relationships–2

Each passenger (customer) of a flight bought a single ticket via a single airline.

Answer: This is not a key constraint

We can make Passenger a entity!



The basics: entities, relationships, and constraints

We covered enough to model *most typical* situations.
We have already seen situations we could not model, however!

Advanced modeling features

- Weak entities.
- ISA Hierarchies.

The need for weak entities

Definition

A *weak entity* is ‘owned by’ another identity:
A weak entity can only be uniquely identified in conjunction with owning entities.

The need for weak entities

Definition

A *weak entity* is ‘owned by’ another identity:
A weak entity can only be uniquely identified in conjunction with owning entities.

Examples

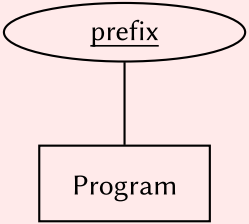
Assignments belong to courses:

- Many courses have a 1-st assignment.
- The 1-st assignment of COMPSCI 2DB3: *ER data model*.

Courses belong to a degree program (at McMaster):

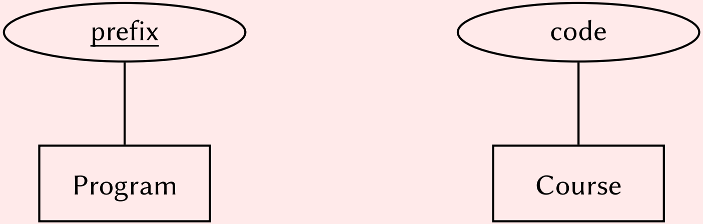
- There are several courses with code 2DB3 (e.g., COMPSCI 2DB3 versus SFWRENG 2DB3).
- There is only one COMPSCI 2DB3 (this course).

Weak entities in practice



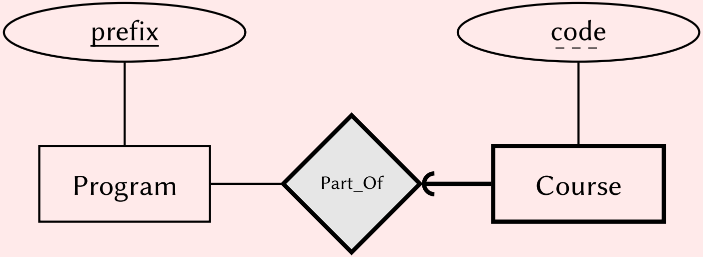
A degree program is an entity with a *unique prefix* (COMPSCI, SFWRENG, …).

Weak entities in practice



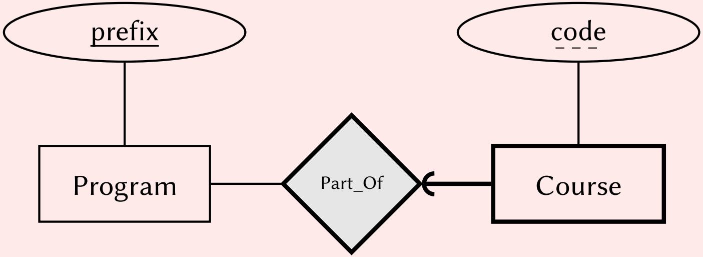
A course has a a *unique code* within a degree program (2DB3 within COMPSCI).

Weak entities in practice



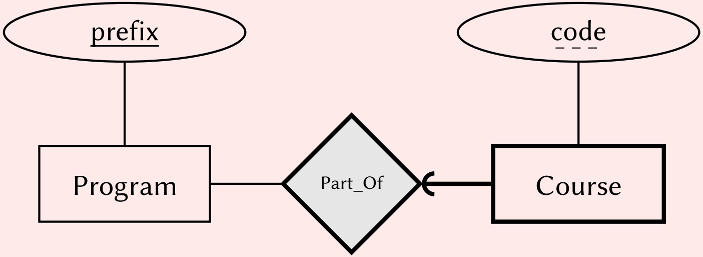
Course is a *weak entity* and Part_Of is an *identifying relationship* (bold).

Weak entities in practice



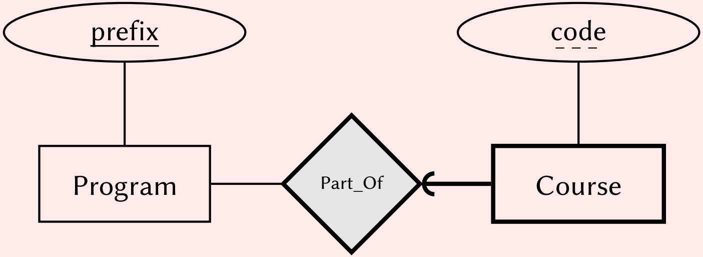
Weak entities must participate in their identifying relationship(s): ⟶.

Weak entities in practice



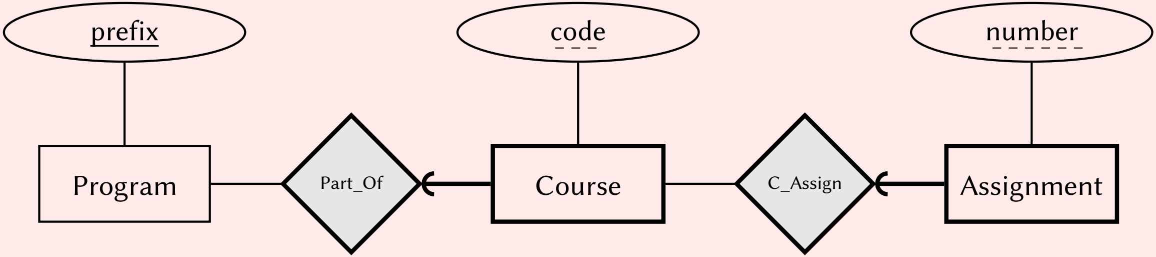
Attribute code is a *partial key* for the weak entity Course.

Weak entities in practice



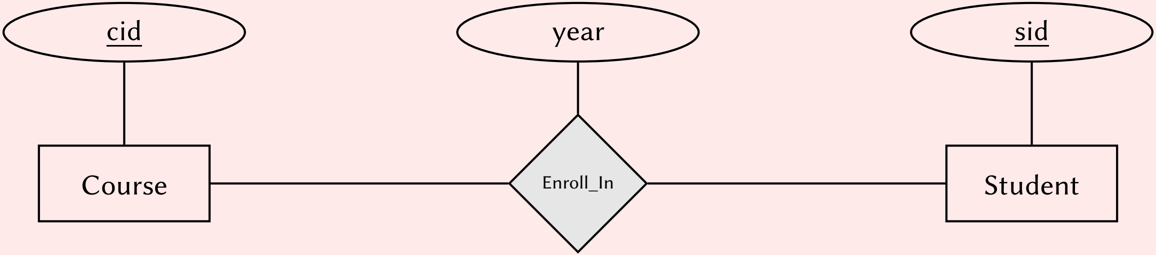
The primary key for weak entity Course is the pair (prefix, code)!

Weak entities in practice



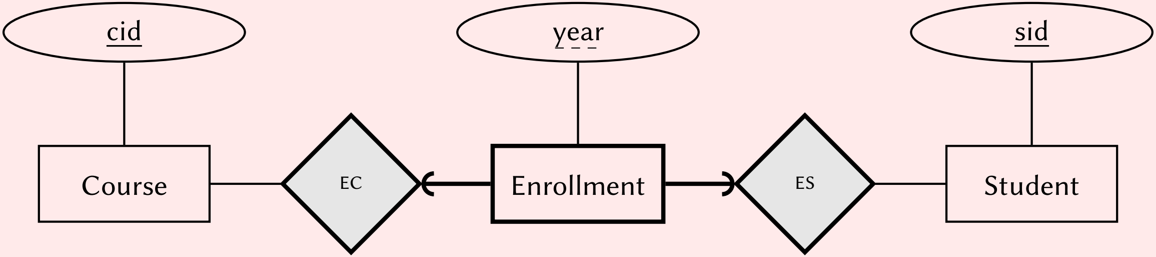
The primary key for weak entity Assignment is the triple (prefix, code, number).

A step-by-step example: Continued



Problem: Student could only enroll once

A step-by-step example: Continued

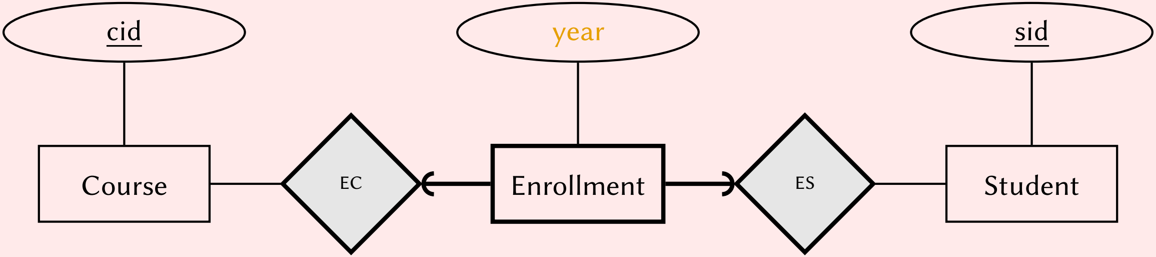


Problem: Student could only enroll once

Turn relationship Enroll_In into a weak entity Enrollment.

- Enrollment has partial key year.
- Enrollment is owned by both a Student and a Course.
- The primary key of Enrollment is the triple (sid, cid, year).

A step-by-step example: Continued

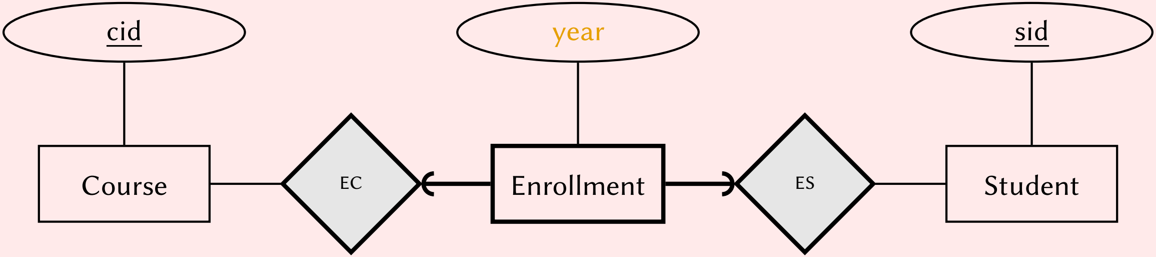


A word of caution

That something *can* be written, does not mean it *should* be written.

Do *not* use weak entities when simpler constructs suffice!

A step-by-step example: Continued



A word of caution

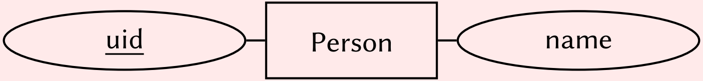
That something *can* be written, does not mean it *should* be written.
Do *not* use weak entities when simpler constructs suffice!

E.g., do not use weak entities to express normal relationships.

Entity modularity via ISAs: Students, Faculty, Staff

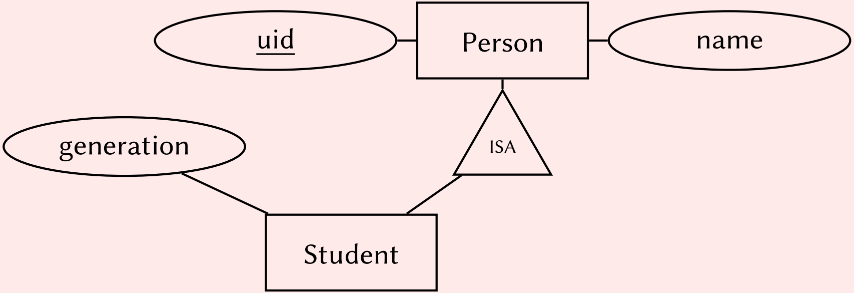
This is a classic example of a *class hierarchy*.

Entity modularity via ISAs: Students, Faculty, Staff



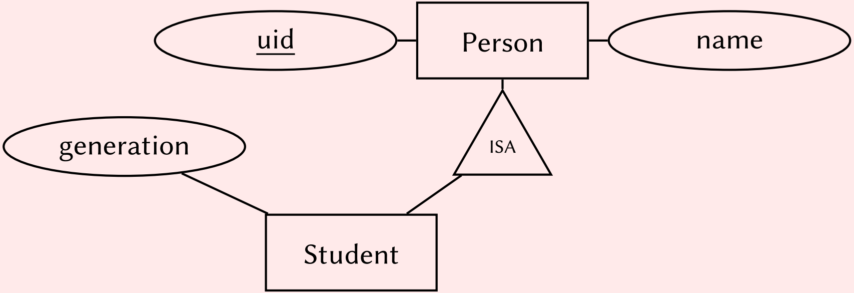
Students, Faculty, and Staff are all *peoples* with names and University IDs.

Entity modularity via ISAs: Students, Faculty, Staff



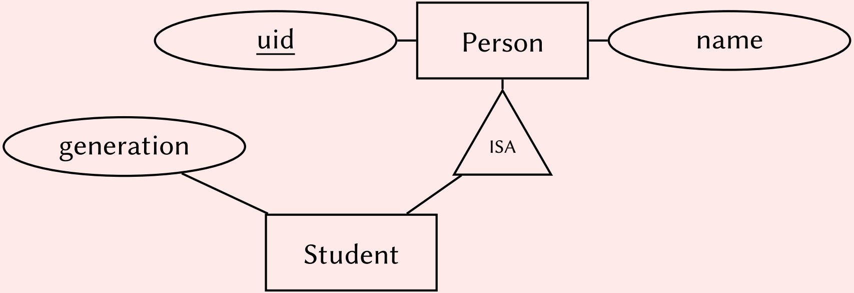
Each student is part of a *generation* (e.g., Class of 2021).

Entity modularity via ISAs: Students, Faculty, Staff



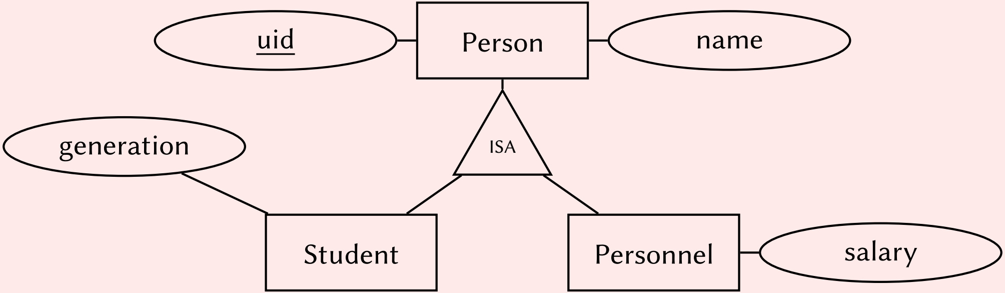
We say that entity Student is a *specialization* or *subclass* of entity Person.

Entity modularity via ISAs: Students, Faculty, Staff



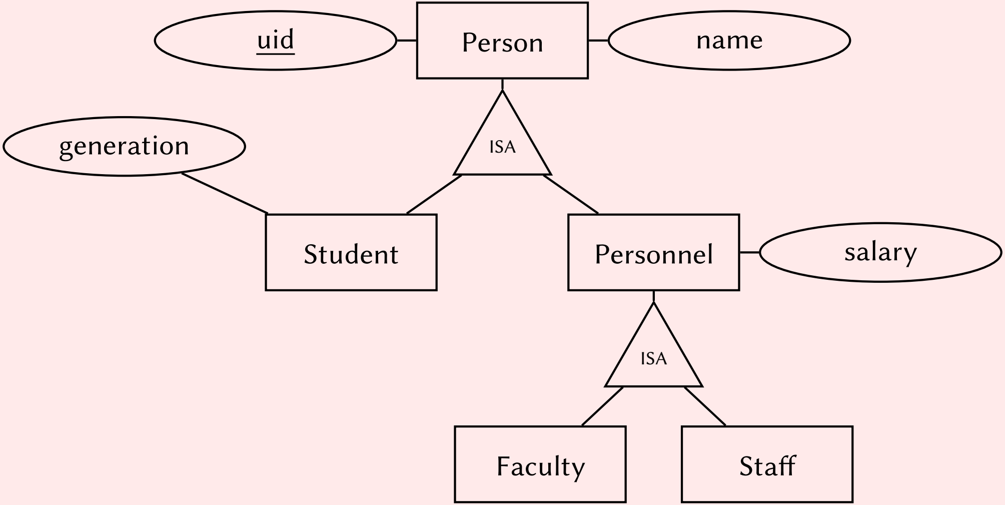
We say that entity Person is a *generalization* or *superclass* of Student.

Entity modularity via ISAs: Students, Faculty, Staff



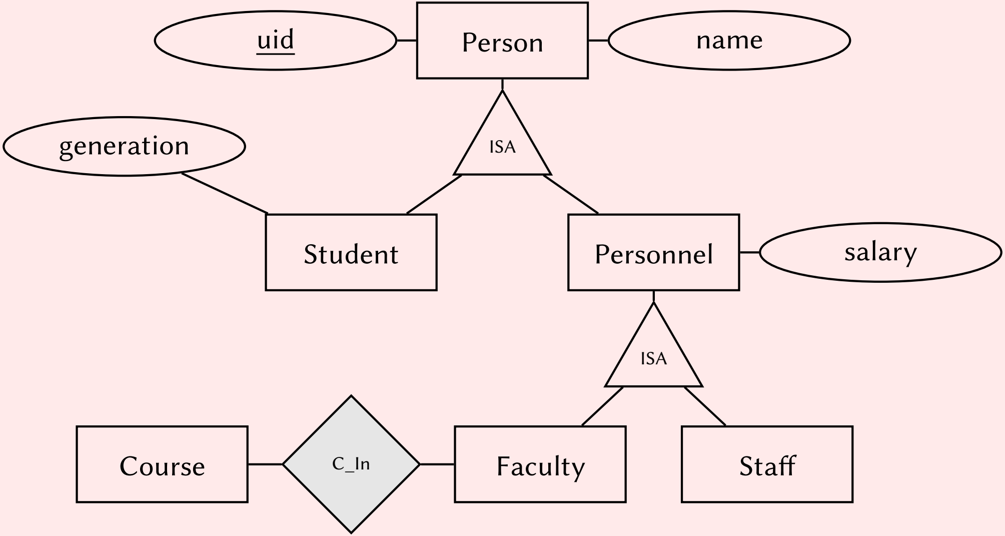
Each faculty and staff member is personnel with a *salary*.

Entity modularity via ISAs: Students, Faculty, Staff



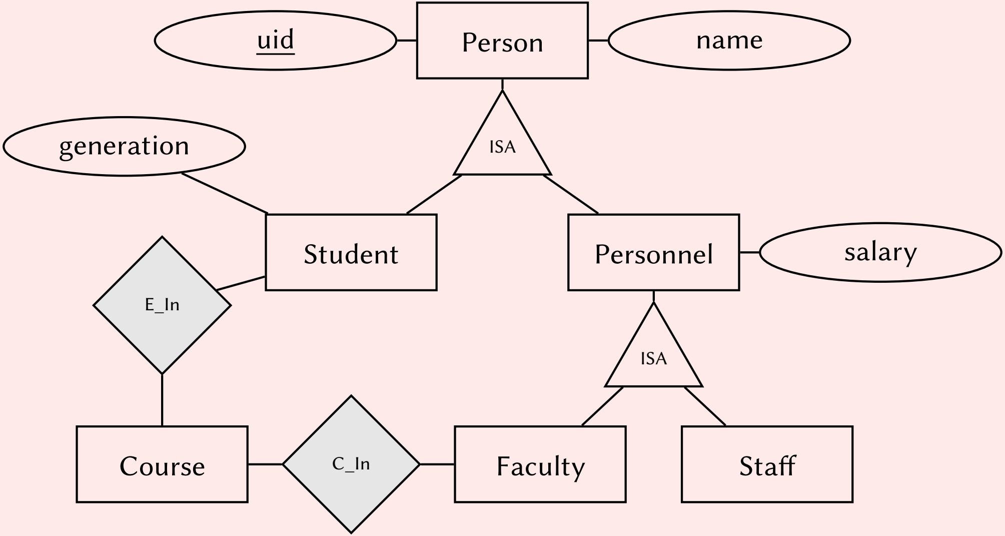
We can further specialize faculty and staff.

Entity modularity via ISAs: Students, Faculty, Staff



Only faculty members are instructors of courses.

Entity modularity via ISAs: Students, Faculty, Staff



While only students can enroll in these courses.

ISA hierarchies require documentation

Every combination of entities in the hierarchy can exists. E.g.,

- there can be Person entities that are not students, faculty, or staff;
- there can be Faculty entities that are also Staff and Students, ...

ISA hierarchies require documentation

Every combination of entities in the hierarchy can exists. E.g.,

- there can be Person entities that are not students, faculty, or staff;
- there can be Faculty entities that are also Staff and Students, ...

Terminology

Overlap constraints Can a single entity belong to *multiple* subclasses?
E.g., can an entity be both Faculty and Student?

Covering constraints Must every superclass entity *also belong* to one of its subclasses?
E.g., must every Person entity be a Student, Faculty, and/or Staff?

ISA hierarchies require documentation

Every combination of entities in the hierarchy can exists. E.g.,

- there can be Person entities that are not students, faculty, or staff;
- there can be Faculty entities that are also Staff and Students, ...

Terminology

Overlap constraints Can a single entity belong to *multiple* subclasses?
E.g., can an entity be both Faculty and Student?

Covering constraints Must every superclass entity *also belong* to one of its subclasses?
E.g., must every Person entity be a Student, Faculty, and/or Staff?

Document all constraints.

On good design: Attributes versus Entities

Attributes should represent a *single atomic* value.
Atomic here means a base value without internal structure:
No lists, complex objects, ...

Entities represent *complex objects*: multiple attributes.

Example: Phone numbers

If you want to model a Person entity with

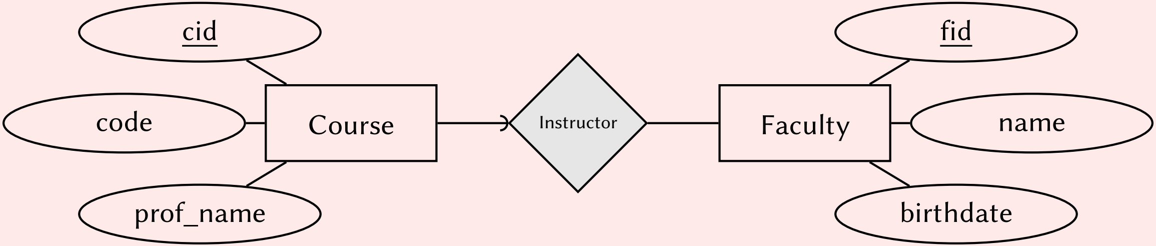
- a *single phone number*: use an attribute.
- *multiple phone numbers*: use a phone number (weak) entity.

On good design: Redundancy

What *redundancy*: storing the same information in multiple ways.

Why Redundancies cause *inconsistencies* during updates.
Redundancies *waste* storage space.

Example of Redundancy



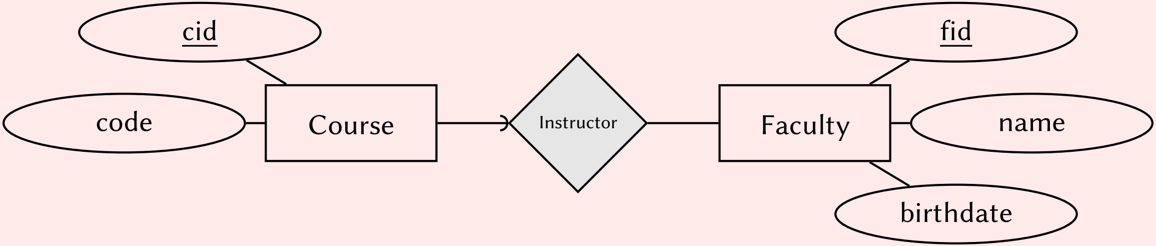
The instructor name is stored *twice*.

On good design: Redundancy

What *redundancy*: storing the same information in multiple ways.

Why Redundancies cause *inconsistencies* during updates.
Redundancies *waste* storage space.

Example of Redundancy (removed)



On good design: Keys & weak entities

- Real-world entities might not have a clear *key*.
  E.g., not everyone has a government-provided SIN.

- We often create unique internal identifiers for entity sets.
  E.g., Mac ID as used within McMaster University.
  E.g., sequential counters in many systems.

Weak entities and normal entities

- Every weak entity can be turned into a normal entity.
- Use weak entities *only* if there there is a clear identifier in the context of an *owner*.
  E.g., the *course code* '2DB3' is identifying a course if you know the program.
  E.g., the *shirt number* '5' is identifying a player if you know the sport team.

Summary

Entity-relationship modeling

- yields a *conceptual schema*: high-level description of the data in terms of:
  - entities, attributes, and relationships,
  - weak entities, ISA hierarchies, and aggregation,
  - constraints: keys, relationship participation, ISA hierarchies;
- depends on a complete *requirements analysis*;
- is highly subjective: many ways to model the same data.

Later lectures: Translating an ER-diagram into tables.