Design Principles

Stay faithful to the application.
- choose entity sets, relationships, attributes, and constraints that make sense for the application
- don't over- or under-constrain

Avoid redundancy.
- store each piece of info in only one place
- avoid potential for update and deletion anomalies

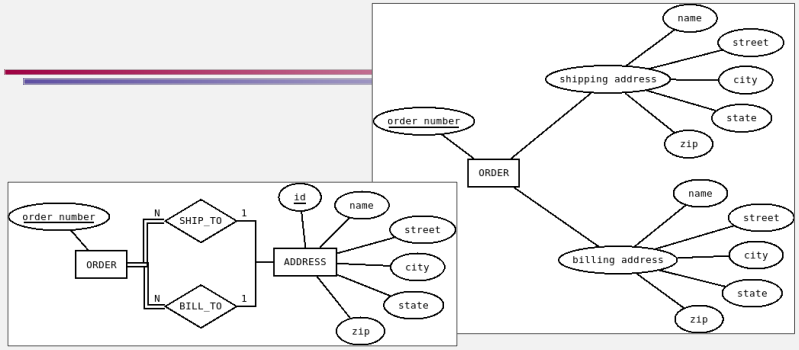
consider redundancy between entities as well as between entity/relationship types

Design Principles

Choose the right relationships.
- don't include relationships that can be deduced from other ones

Choose the right kind of elements.
- promote an attribute to a relationship if it refers to another entity type
- promote an attribute to an entity type if it is present in multiple entity types, especially if it is a composite attribute
- demote an entity with only one attribute and a single relationship to an attribute

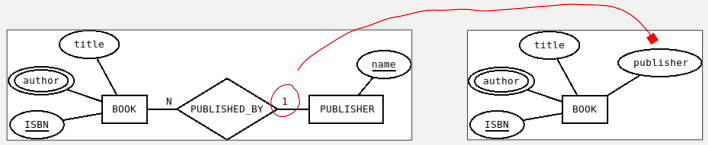
Strive for simplicity.
- don't include more elements than necessary
- don't create entity sets when attributes would do
- don't create entity sets when a relationship would do



Promote a composite attribute appearing in more than one place to an entity type.	3 respondents	**60 %** ✓
Use only the key to refer to another entity.	2 respondents	40 %
Use a relationship instead of an attribute that refers to another entity.	3 respondents	60 %
Demote an entity type with only one attribute and one relationship to an attribute.		0 %

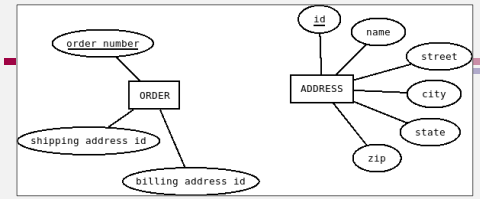
promote composite attributes appearing in more than one place to an entity type
- e.g. "address" appears both as a shipping address and a billing address
  → redundant to express the components of an address multiple times
  → also redundant storage if the same address appears as both

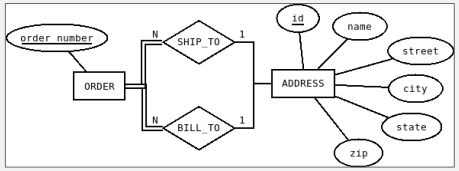
cardinality 1 → regular single-valued attribute
cardinality N → multivalued attribute



Promote a composite attribute appearing in more than one place to an entity type.	1 respondent	20 %
Use only the key to refer to another entity.		0 %
Use a relationship instead of an attribute that refers to another entity.	3 respondents	60 %
Demote an entity type with only one attribute and one relationship to an attribute.	2 respondents	**40 %** ✓

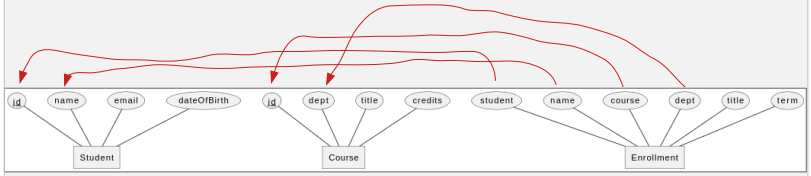
demote entity types with only one attribute and relationship to attributes
- keep it simple – entity type + relationship is more complex and doesn't add anything

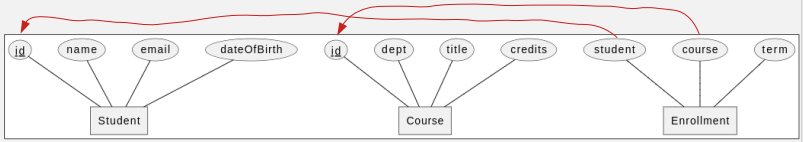




Promote a composite attribute appearing in more than one place to an entity type.	1 respondent	20 %
Use only the key to refer to another entity.	1 respondent	20 %
Use a relationship instead of an attribute that refers to another entity.	3 respondents	60 % ✓
Demote an entity type with only one attribute and one relationship to an attribute.	1 respondent	20 %

Use only the key to refer to another entity.





Design Decisions – Constraints

To include or not include a constraint (key, NOT NULL, participation, cardinality) –

- Is the constraint an inherent property of the data, or a particular rule imposed by the application (and which might change)?
  - e.g. there cannot be two players with the same number on a team vs a student registers for at most four courses in a term
- Is the constraint only applicable at certain points in time?
  - e.g. a student must have a major and a minor to graduate but not at every point of their college career
- Keep in mind the possibility that data might exist but be unknown.
  - e.g. every movie has to be directed by someone, but we might not know who that director is

Would security or other considerations require more constraints on relationships?

- goal of conceptual modeling is to capture structural relationships between data elements
  - (may be modeled as entity with attribute, composite attribute, relationship between entity types, …)
- security deals with who is allowed access to the data
  - captured in the conceptual model only to the degree that data is needed to support access determination
    - e.g. store usernames and passwords for determining access to application features
    - RDBMS account and permissions system is the basis for determining access to data