Module 7: The Entity Relationship Data Model

Winter 2026

Cheriton School of Computer Science

CS 348: Intro to Database Management

Reading Assignments and References

To be read during the Week of February 23–27:

- Chapter 6 of course textbook.[1] (Material in Sections 6.7 will be covered in the next module.)

[1] Silberschatz, Korth and Sudarshan, *Database Systems Concepts*, 7th edition

Outline

Overview of the ER Data Model (ERM)

Conceptual Data Model

A *conceptual data model* serves as a first step in formally capturing the metadata for information systems. Informal requirements for the underlying information are mapped to such a model.

ERM is a conceptual data model proposed in a paper by Peter Chen in 1976. A collection of metadata expressed in ERM is referred to as an *ER model*.

Overview of the ER Data Model (ERM)

Conceptual Data Model

A *conceptual data model* serves as a first step in formally capturing the metadata for information systems. Informal requirements for the underlying information are mapped to such a model.

ERM is a conceptual data model proposed in a paper by Peter Chen in 1976. A collection of metadata expressed in ERM is referred to as an *ER model*.

Metadata is formally captured in terms of

- entities,
- attributes, and
- relationships.

Overview of the ER Data Model (ERM)

Conceptual Data Model

A *conceptual data model* serves as a first step in formally capturing the metadata for information systems. Informal requirements for the underlying information are mapped to such a model.

ERM is a conceptual data model proposed in a paper by Peter Chen in 1976. A collection of metadata expressed in ERM is referred to as an *ER model*.

Metadata is formally captured in terms of

- entities,
- attributes, and
- relationships.

ER Diagram

The syntax for specifying an ER model is in the form of a graphical visualization.

There are *many* dialects of ERM and notational conventions for ER diagrams.

ER Modeling: Entities

Entity A *distinguishable* thing.

Entity Set A set of entities of the same variety.

Examples of entity sets:

1. *students currently at University of Waterloo*
2. *flights offered by Air Canada*
3. *burglaries in Ontario during 1994*

Graphical visualization:

ER Modeling: Attributes

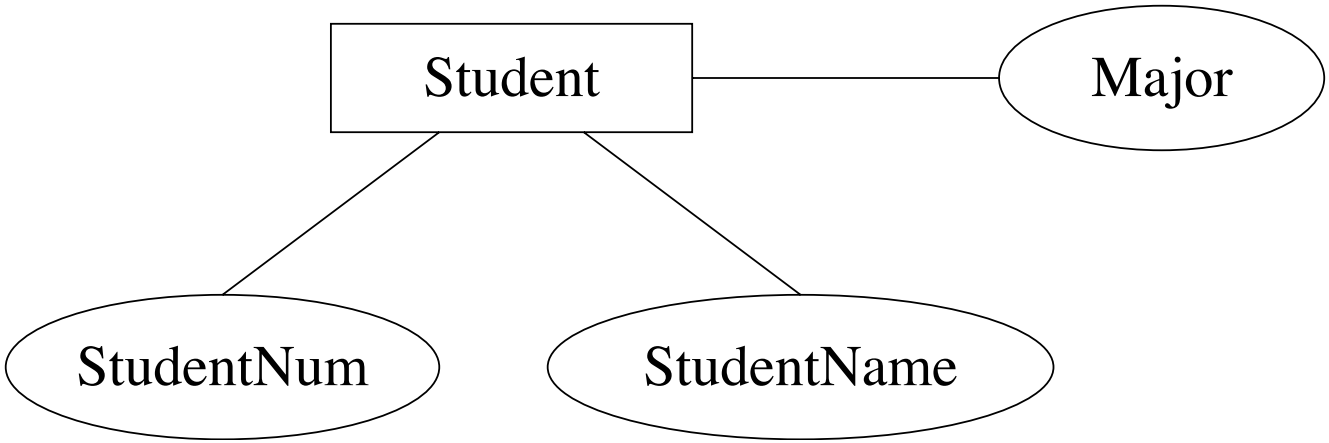
Attribute Captures a concrete fact about an entity.

Domain A set of possible values for an attribute.

Examples of attributes for entities that are students:

1. *student number*
2. *student name*
3. *major*

Graphical visualization:



ER Modeling: Relationships

Relationship Captures the existence of an association between two or more entities.

Relationship Set A set of relationships of the same variety.

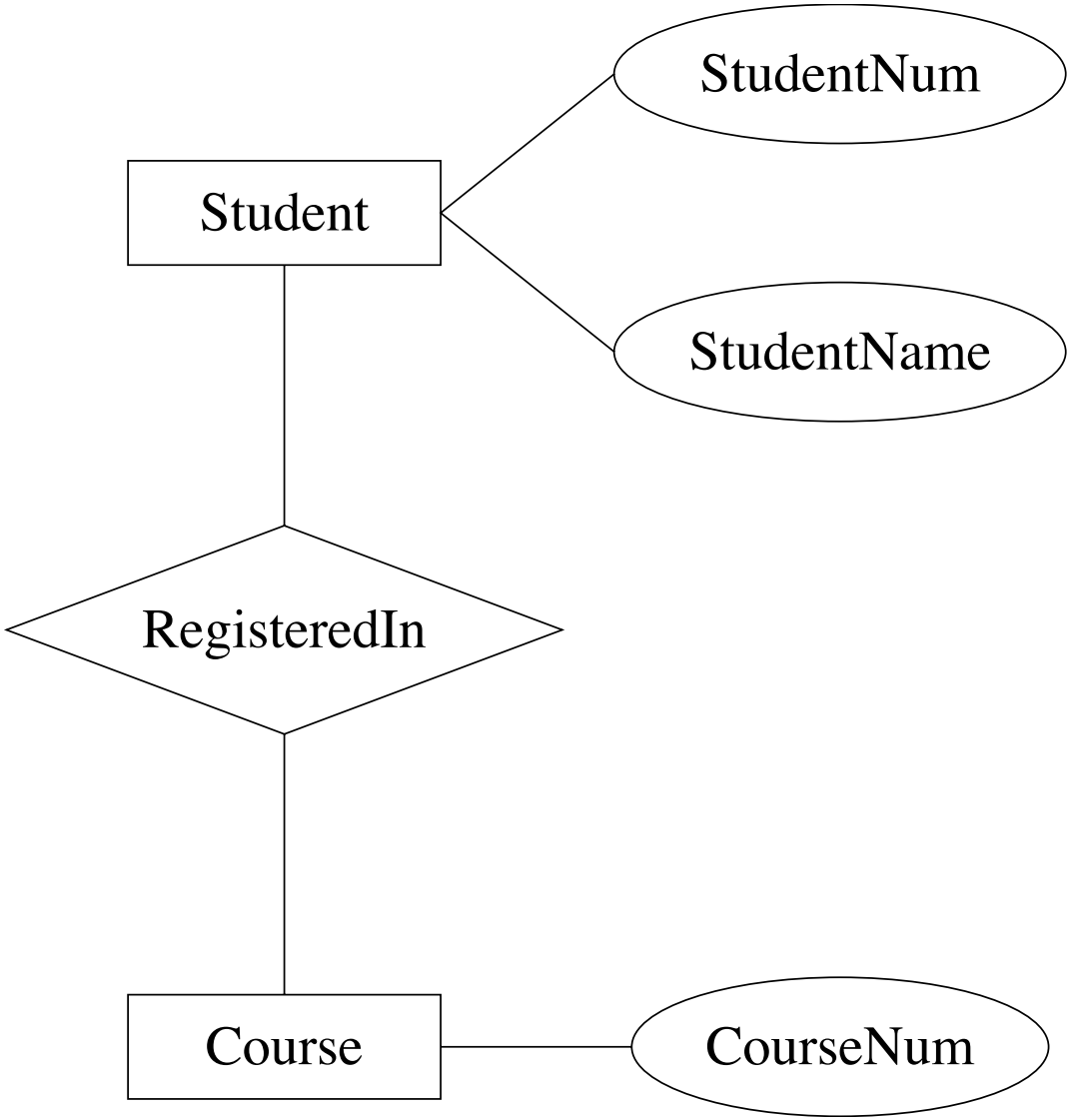
Examples of relationship sets:

1. *students registered in courses*
2. *bank branches, customers and their accounts*
3. *passengers booked on flights*
4. *parents and their children*

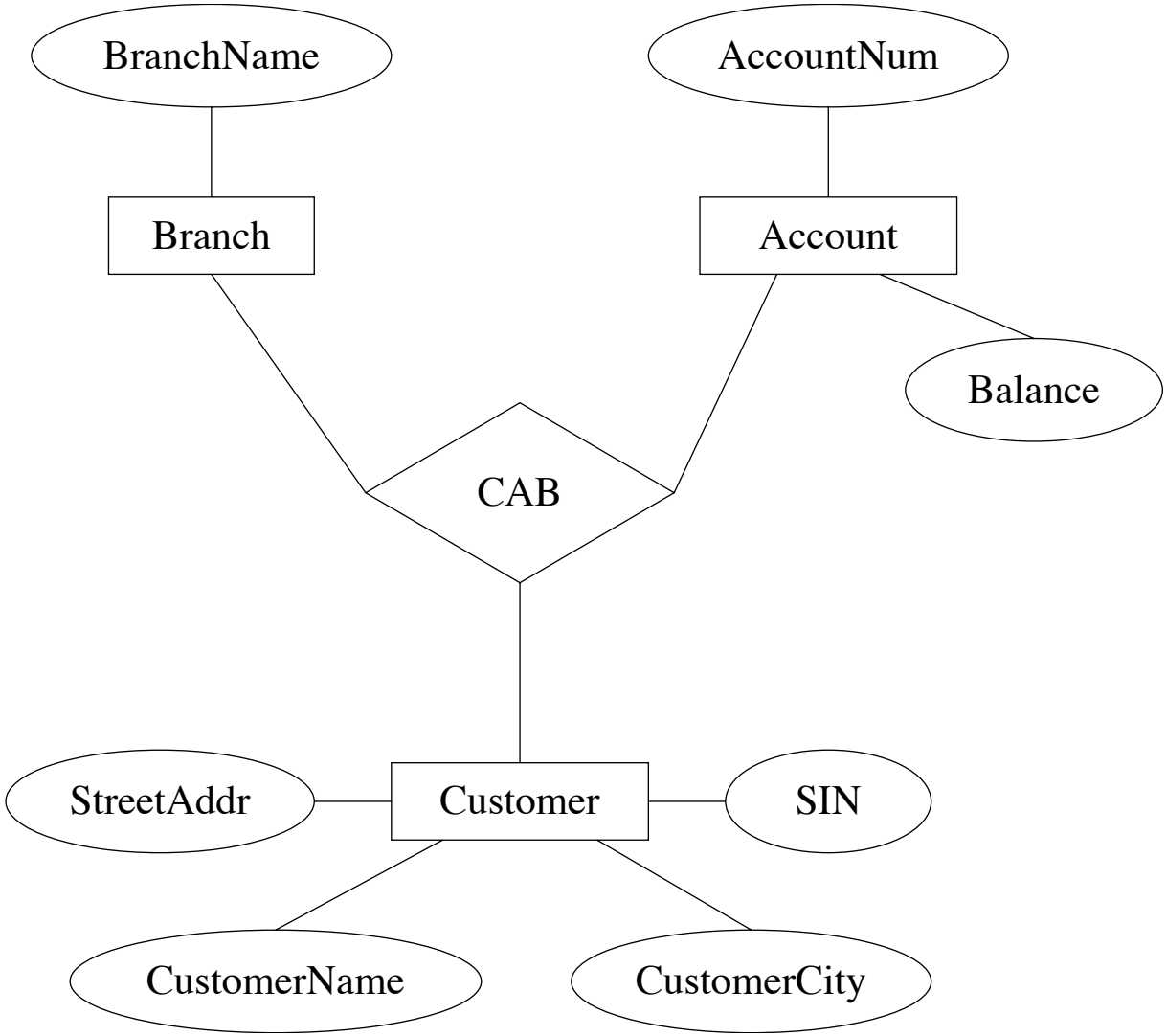
Graphical visualizations of the first two examples follow.

Note: Relationships are uniquely determined by their participating entities.

ER Modeling: Relationships (cont'd)



ER Modeling: Relationships (cont'd)



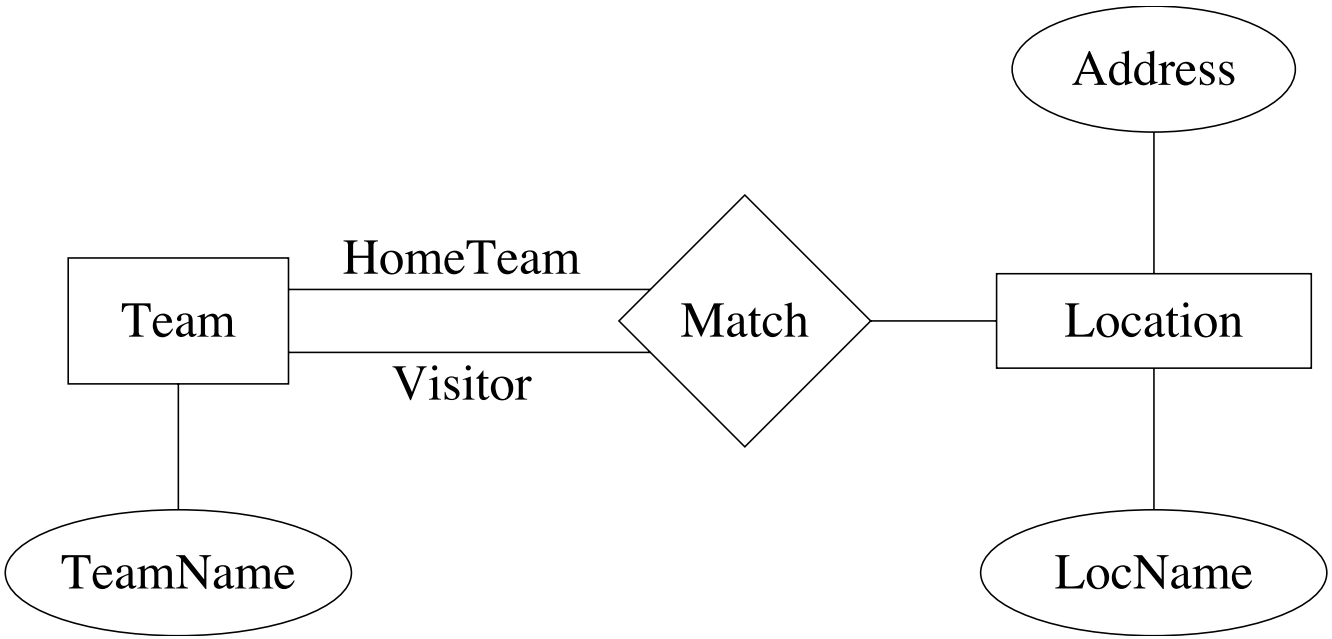
Multiple Entity Roles and Role Names

Role The purpose served by a particular entity in a relationship.

Role Name An identifier indicative of this purpose.

Examples of role names: *A match takes place between a home team and a visitor.*

Graphical visualization:



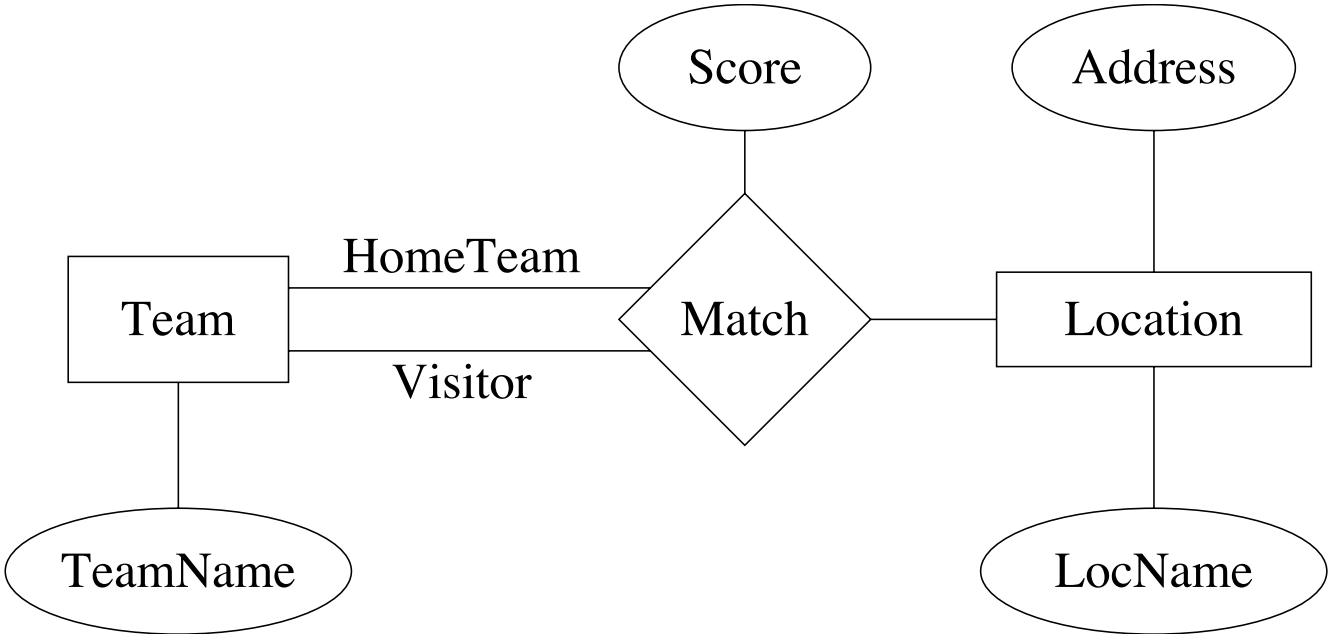
Note: *Explicit* role names are needed whenever a component entity set of a relationship set serves more than one purpose in its relationships.

ER Modeling: Relationships (cont'd)

Relationships may also have attributes.

Example: *A match has a score.*

Graphical visualization:



Note: Relationships are *still* uniquely determined by their participating entities.

Outline

Constraints in an ER Model

There are four varieties of integrity constraints in an ER model that are commonly expressed with graphical annotations:

- primary keys
- binary relationship types
- existence dependencies
- general cardinality constraints

Constraints in an ER Model

There are four varieties of integrity constraints in an ER model that are commonly expressed with graphical annotations:

- primary keys
- binary relationship types
- existence dependencies
- general cardinality constraints

General integrity constraints can be captured as sentences in the relational calculus by defining a *mapping* of ER diagrams to relational signatures.†

† Not covered in textbook.

Constraints in an ER Model

There are four varieties of integrity constraints in an ER model that are commonly expressed with graphical annotations:

- primary keys
- binary relationship types
- existence dependencies
- general cardinality constraints

General integrity constraints can be captured as sentences in the relational calculus by defining a *mapping* of ER diagrams to relational signatures.†

Such a mapping also yields both a formal semantics and an ER query language.

† Not covered in textbook.

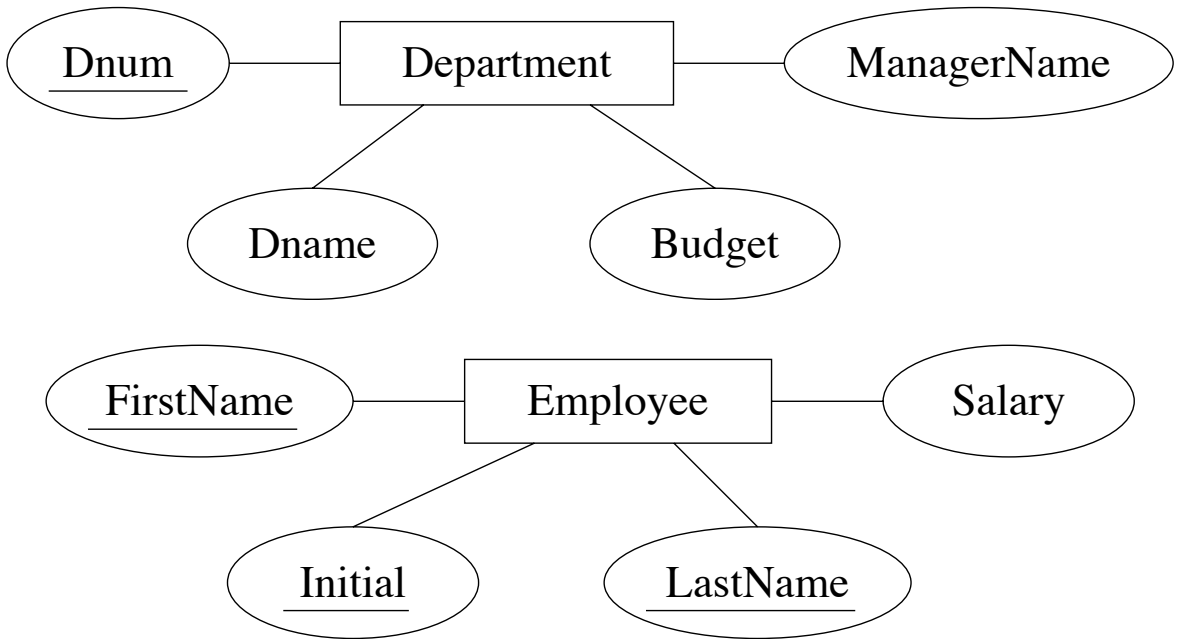
Primary Keys

Primary Key A selection of attributes for an entity set for which facts serve as the means of *reference* to its entities.

Examples:

1. *departments are identified by their department number*
2. *employees are identified by their first name, middle initial and last name*

Graphical annotation (via *underlining*):



Existence Dependencies

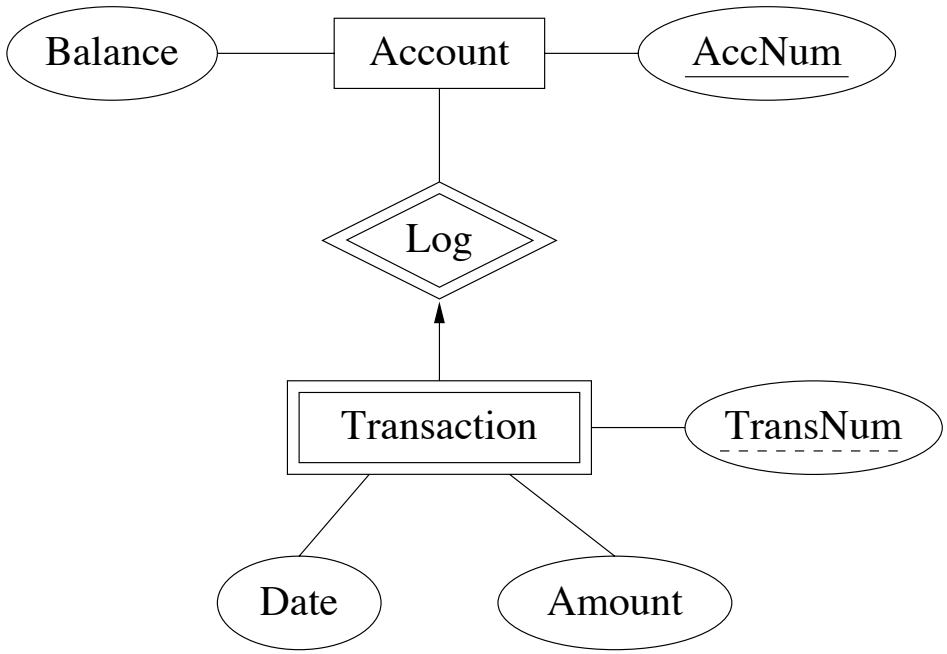
Sometimes the existence of an entity depends on the existence of another entity.

If *x* is existence dependent on *y*, then

1. *y* is a *dominant entity*, and
2. *x* is a *subordinate entity*.

Example: *Transactions are existence dependent on accounts.*

Graphical annotation (via *double boundaries* on an entity set):



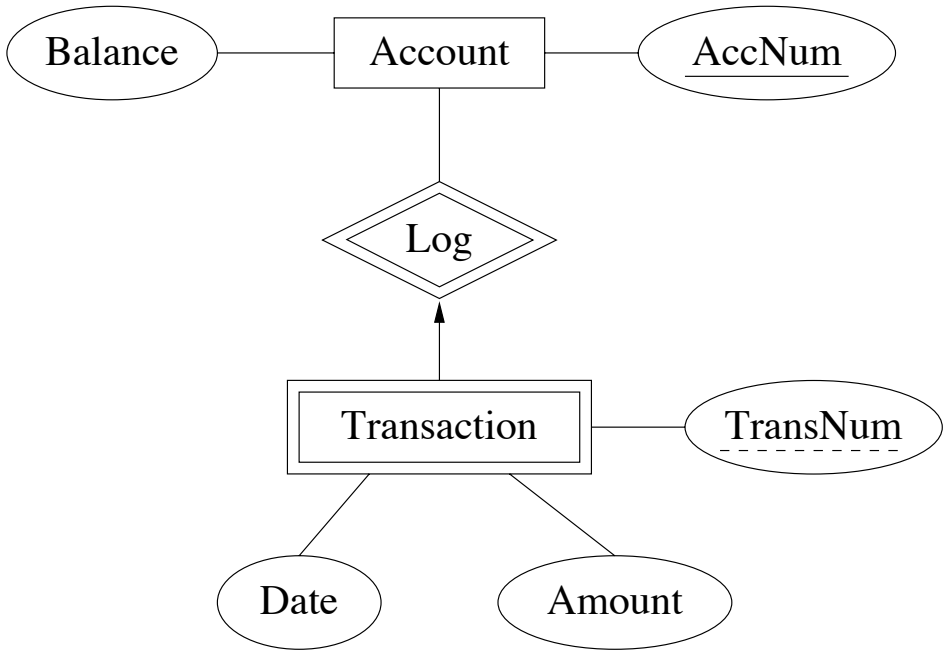
Identifying Subordinate Entities

Weak Entity Set An entity set containing subordinate entities.

Discriminator A selection of attributes for a weak entity set for which facts serve as the means of distinguishing subordinate entities for any given dominant entity.

Example: *Each transaction for a given account has a unique transaction number.*

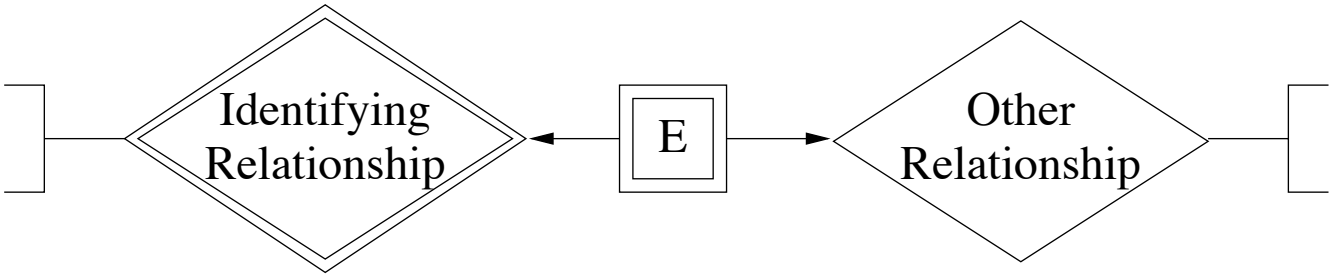
Graphical annotation (via *dashed underlining* of attributes for a weak entity set):



Identifying Subordinate Entities (cont'd)

A weak entity set must be in a (N:1) relationship with at least one distinct entity set.

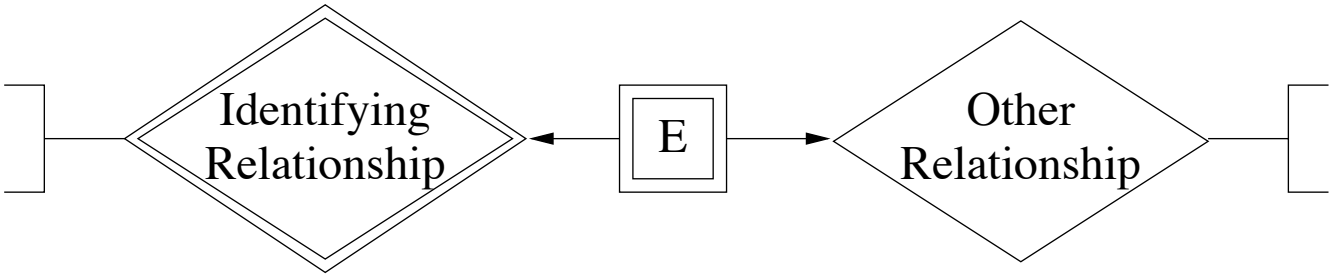
Graphical annotation (via *double boundaries* for identifying relationships):



Identifying Subordinate Entities (cont'd)

A weak entity set must be in a (N:1) relationship with at least one distinct entity set.

Graphical annotation (via *double boundaries* for identifying relationships):



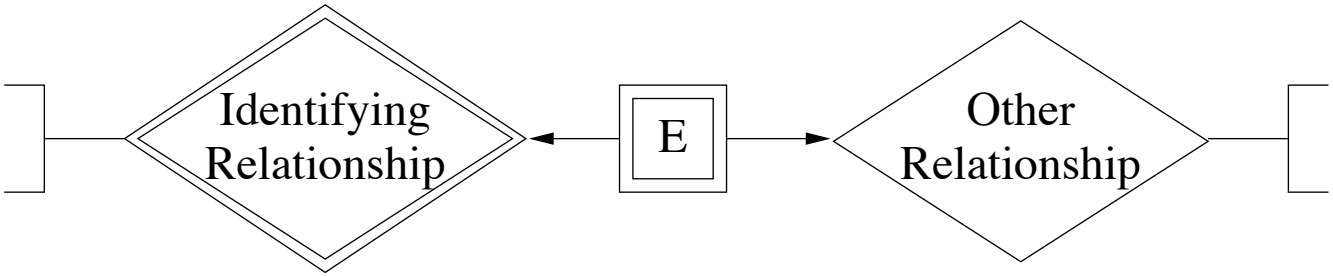
Primary Key of a Weak Entity Set

The means of reference to a subordinant entity of a weak entity set are the facts for its discriminator together with the facts for each attribute of the primary keys of the entity sets for dominant entities.

Identifying Subordinate Entities (cont'd)

A weak entity set must be in a (N:1) relationship with at least one distinct entity set.

Graphical annotation (via *double boundaries* for identifying relationships):



Primary Key of a Weak Entity Set

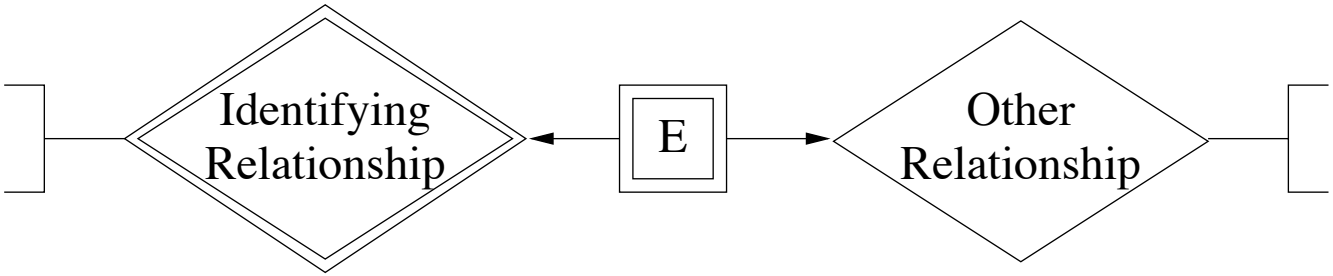
The means of reference to a subordinant entity of a weak entity set are the facts for its discriminator together with the facts for each attribute of the primary keys of the entity sets for dominant entities.

Note: This definition is recursive since entity sets of dominant entities may themselves be weak (a dominant entity may itself be subordinate to another entity).

Identifying Subordinate Entities (cont'd)

A weak entity set must be in a (N:1) relationship with at least one distinct entity set.

Graphical annotation (via *double boundaries* for identifying relationships):



Primary Key of a Weak Entity Set

The means of reference to a subordinant entity of a weak entity set are the facts for its discriminator together with the facts for each attribute of the primary keys of the entity sets for dominant entities.

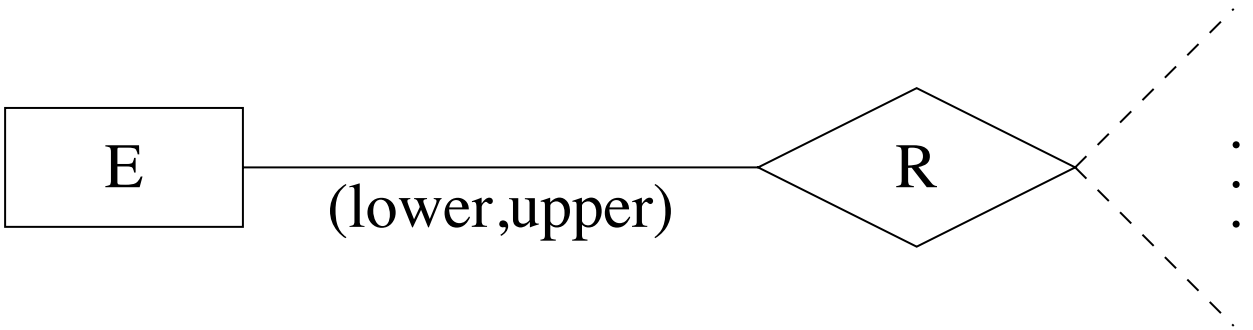
Note: This definition is recursive since entity sets of dominant entities may themselves be weak (a dominant entity may itself be subordinate to another entity).

Note: No cycles of binary relationship sets that are all identifying are allowed in an ER diagram.

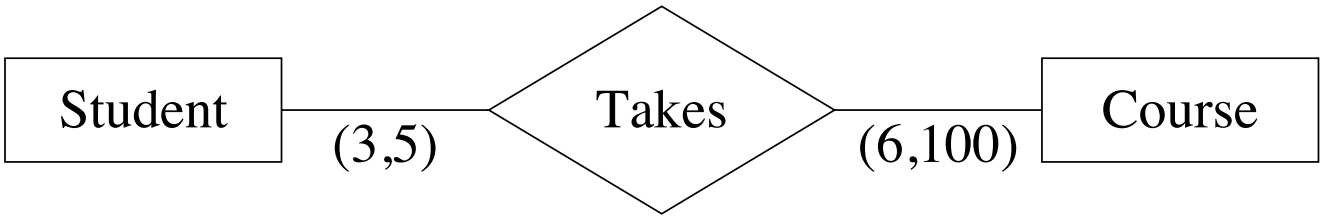
General Cardinality Constraints

A general cardinality constraint determines lower and upper bounds on the number of relationships of a given relationship set in which a component entity must participate.

Graphical annotation (via *component edge labelling*):



Example: *Students must take between 3 and 5 courses; courses must have between 6 and 100 students taking them.*



Note: The upper bound may be “N”, which indicates that no upper bound exists.

Binary Relationship Types

many-to-many (N:N): An entity in one entity set can be related to any number of entities in the other, and the converse also holds.
$\Rightarrow$ relationship RegisteredIn is many-to-many

many-to-one (N:1): Each entity in one entity set can be related to at most one entity in the other entity set, but no such limit exists for the converse.

one-to-many (1:N): Inverse of many-to-one.

one-to-one (1:1): Each entity in one entity set can be related to at most one entity in the other, and the same holds for the converse.

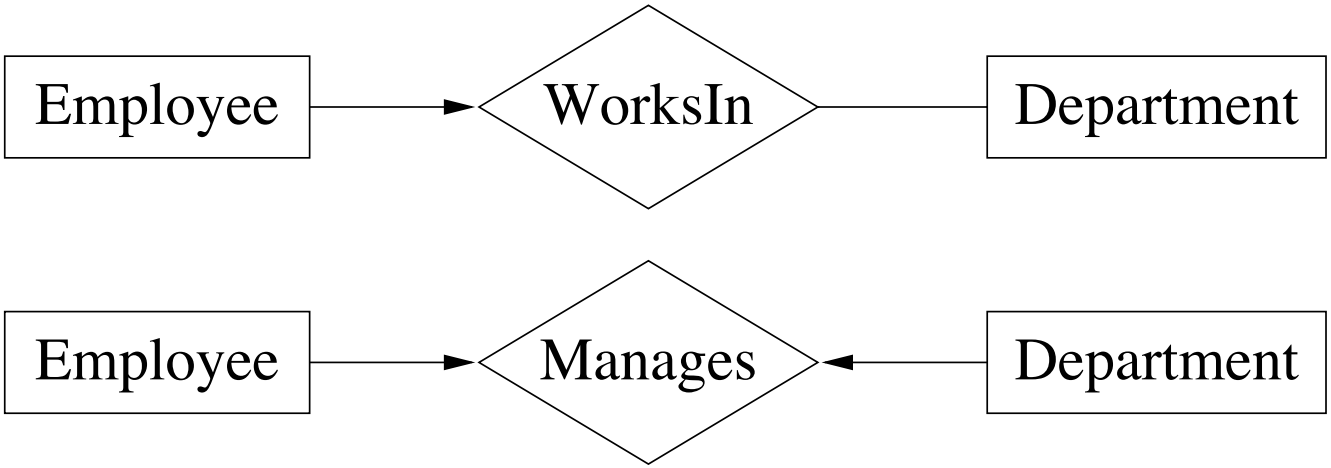
Note: None of these binary relationship types imply any *mandatory* participation of entities.

Binary Relationship Types (cont'd)

Examples:

1. *Employees work in at most one department.*
2. *Employees manage at most one department, and departments are managed by at most one employee.*

Graphical annotation (via *arrowheads*):



General Integrity Constraints

General integrity constraints for an ER diagram can be expressed in the relational calculus via a mapping to a relational signature ρ satisfying:

- `<E>(self)` $\in \rho$, for each entity set named `<E>`;
- `<A>(self,<a>)` $\in \rho$ for each attribute named `<a>`; and
- `<R>(`c_1`,...,`c_k`)` $\in \rho$ for each relationship set for k-ary associations named `<R>`, where c_i is either an entity set name or a role name.

Note that arity is indicated by a sequence of *relational* attributes.

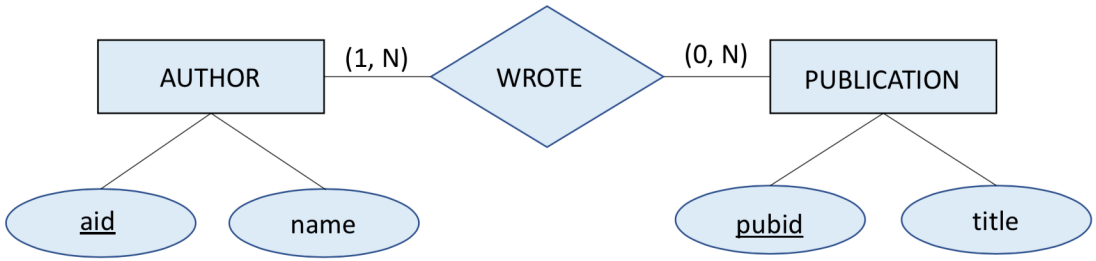
General Integrity Constraints

General integrity constraints for an ER diagram can be expressed in the relational calculus via a mapping to a relational signature ρ satisfying:

- `<E>(self)` $\in \rho$, for each entity set named `<E>`;
- `<A>(self,<a>)` $\in \rho$ for each attribute named `<a>`; and
- `<R>(`c_1`,...,`c_k`)` $\in \rho$ for each relationship set for k-ary associations named `<R>`, where c_i is either an entity set name or a role name.

Note that arity is indicated by a sequence of *relational* attributes.

Example (from Module 1):



$\rho = ($ `AUTHOR(self), AID(self, aid), NAME(self, name),`
`PUBLICATION(self), PUBID(self, pubid), TITLE(self, title),`
`WROTE(author, publication)` $)$

Example (cont'd)

Some integrity constraints induced by the bibleography ER diagram:

- *Attributes* `aid` *and* `name` *are single-valued.*

  $\forall e, v_1, v_2.(\texttt{AID}(e, v_1) \wedge \texttt{AID}(e, v_2) \rightarrow v_1 = v_2)$

  $\forall e, v_1, v_2.(\texttt{NAME}(e, v_1) \wedge \texttt{NAME}(e, v_2) \rightarrow v_1 = v_2)$

Example (cont'd)

Some integrity constraints induced by the bibleography ER diagram:

- *Attributes* `aid` *and* `name` *are single-valued.*

  $\forall e, v_1, v_2.(\texttt{AID}(e, v_1) \wedge \texttt{AID}(e, v_2) \rightarrow v_1 = v_2)$

  $\forall e, v_1, v_2.(\texttt{NAME}(e, v_1) \wedge \texttt{NAME}(e, v_2) \rightarrow v_1 = v_2)$

- *Entities of a given entity set have a given attribute value.*

  $\forall e.(\texttt{AUTHOR}(e) \rightarrow \texttt{AID}(e, -))$

Example (cont'd)

Some integrity constraints induced by the bibleography ER diagram:

- *Attributes* `aid` *and* `name` *are single-valued.*

  $\forall e, v_1, v_2.(\texttt{AID}(e, v_1) \wedge \texttt{AID}(e, v_2) \rightarrow v_1 = v_2)$

  $\forall e, v_1, v_2.(\texttt{NAME}(e, v_1) \wedge \texttt{NAME}(e, v_2) \rightarrow v_1 = v_2)$

- *Entities of a given entity set have a given attribute value.*

  $\forall e.(\texttt{AUTHOR}(e) \rightarrow \texttt{AID}(e, -))$

- *Participating entities of a* `WROTE` *relationship must be an author and a publication.*

  $\forall e.(\texttt{WROTE}(e, -) \rightarrow \texttt{AUTHOR}(e))$

  $\forall e.(\texttt{WROTE}(-, e) \rightarrow \texttt{PUBLICATION}(e))$.

Example (cont'd)

Some integrity constraints induced by the bibleography ER diagram:

- *Attributes* `aid` *and* `name` *are single-valued.*

  $\forall e, v_1, v_2.(\texttt{AID}(e, v_1) \wedge \texttt{AID}(e, v_2) \rightarrow v_1 = v_2)$

  $\forall e, v_1, v_2.(\texttt{NAME}(e, v_1) \wedge \texttt{NAME}(e, v_2) \rightarrow v_1 = v_2)$

- *Entities of a given entity set have a given attribute value.*

  $\forall e.(\texttt{AUTHOR}(e) \rightarrow \texttt{AID}(e, -))$

- *Participating entities of a* `WROTE` *relationship must be an author and a publication.*

  $\forall e.(\texttt{WROTE}(e, -) \rightarrow \texttt{AUTHOR}(e))$

  $\forall e.(\texttt{WROTE}(-, e) \rightarrow \texttt{PUBLICATION}(e))$.

- *Values for attribute* `aid` *are used to identity authors.*

  $\forall e_1, e_2, v.(\texttt{AUTHOR}(e_1) \wedge \texttt{AID}(e_1, v) \wedge \texttt{AUTHOR}(e_2) \wedge \texttt{AID}(e_2, v) \rightarrow e_1 = e_2)$

Example (cont'd)

Some integrity constraints induced by the bibleography ER diagram:

- *Attributes* `aid` *and* `name` *are single-valued.*

  $\forall e, v_1, v_2.(\texttt{AID}(e, v_1) \wedge \texttt{AID}(e, v_2) \rightarrow v_1 = v_2)$

  $\forall e, v_1, v_2.(\texttt{NAME}(e, v_1) \wedge \texttt{NAME}(e, v_2) \rightarrow v_1 = v_2)$

- *Entities of a given entity set have a given attribute value.*

  $\forall e.(\texttt{AUTHOR}(e) \rightarrow \texttt{AID}(e, -))$

- *Participating entities of a* `WROTE` *relationship must be an author and a publication.*

  $\forall e.(\texttt{WROTE}(e, -) \rightarrow \texttt{AUTHOR}(e))$

  $\forall e.(\texttt{WROTE}(-, e) \rightarrow \texttt{PUBLICATION}(e)).$

- *Values for attribute* `aid` *are used to identity authors.*

  $\forall e_1, e_2, v.(\texttt{AUTHOR}(e_1) \wedge \texttt{AID}(e_1, v) \wedge \texttt{AUTHOR}(e_2) \wedge \texttt{AID}(e_2, v) \rightarrow e_1 = e_2)$

- *An author must have written at least one publication.*

  $\forall e.(\texttt{AUTHOR}(e) \rightarrow \texttt{WROTE}(e, -))$

Exercises

1. Write integrity constraints for each of the following:
   1.1 *No two authors have the same name.*
   1.2 *Mary has authored at least two publications.*
   1.3 *John has always had a coauthor.*

Exercises

1. Write integrity constraints for each of the following:
   1.1 *No two authors have the same name.*
   1.2 *Mary has authored at least two publications.*
   1.3 *John has always had a coauthor.*

2. Remember that relationships may also have attributes.
   2.1 *How should our mapping to a relational signature be revised to accommodate this?*

   Hint: Consider reification (illustrated in next unit).

   2.2 *What additional integrity constraints would then be induced?*

Exercises

1. Write integrity constraints for each of the following:
   1.1 *No two authors have the same name.*
   1.2 *Mary has authored at least two publications.*
   1.3 *John has always had a coauthor.*

2. Remember that relationships may also have attributes.
   2.1 *How should our mapping to a relational signature be revised to accommodate this?*

   Hint: Consider reification (illustrated in next unit).

   2.2 *What additional integrity constraints would then be induced?*

3. Remember that a relational database schema begins with a signature.
   3.1 *What problems might there be in considering an ER diagram to also be a relational database schema defined by this mapping?*

   Hint: Consider the purpose of *primary keys* and what this implies about the assumption of constants existing for *every* value in a universe of an RDB instance.

Outline

EER Modelling: Additional Features

- Structured Attributes
- Aggregation
- Specialization
- Generalization
- Disjointness

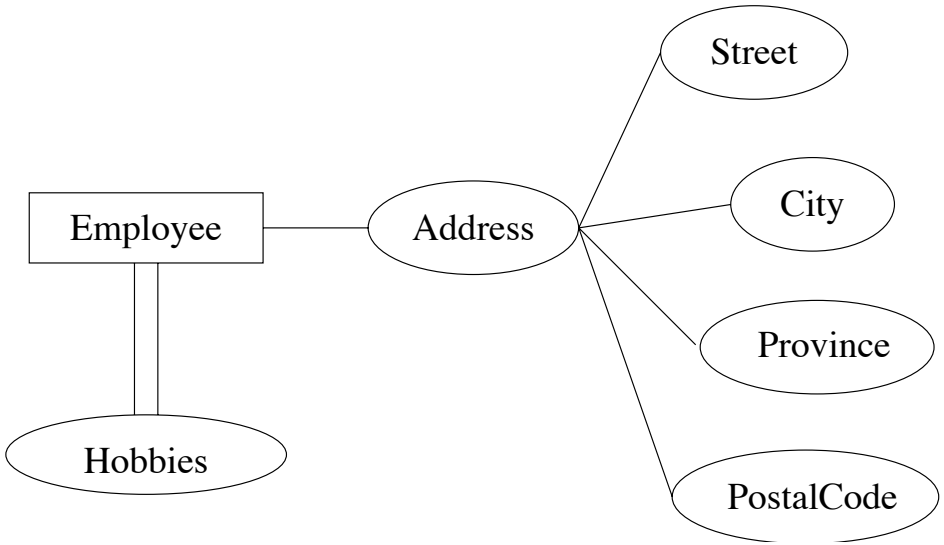
Structured Attributes

Composite Attribute Denotes a fixed collection of other attribute facts.

Multi-Valued Attribute Denotes a finite set of similar facts.

Example: `Hobbies` *are a set of facts about leisure activities; an* `Address` *consists of a* `Street`, `City`, `Province` *and* `PostalCode`.

Graphical visualization:



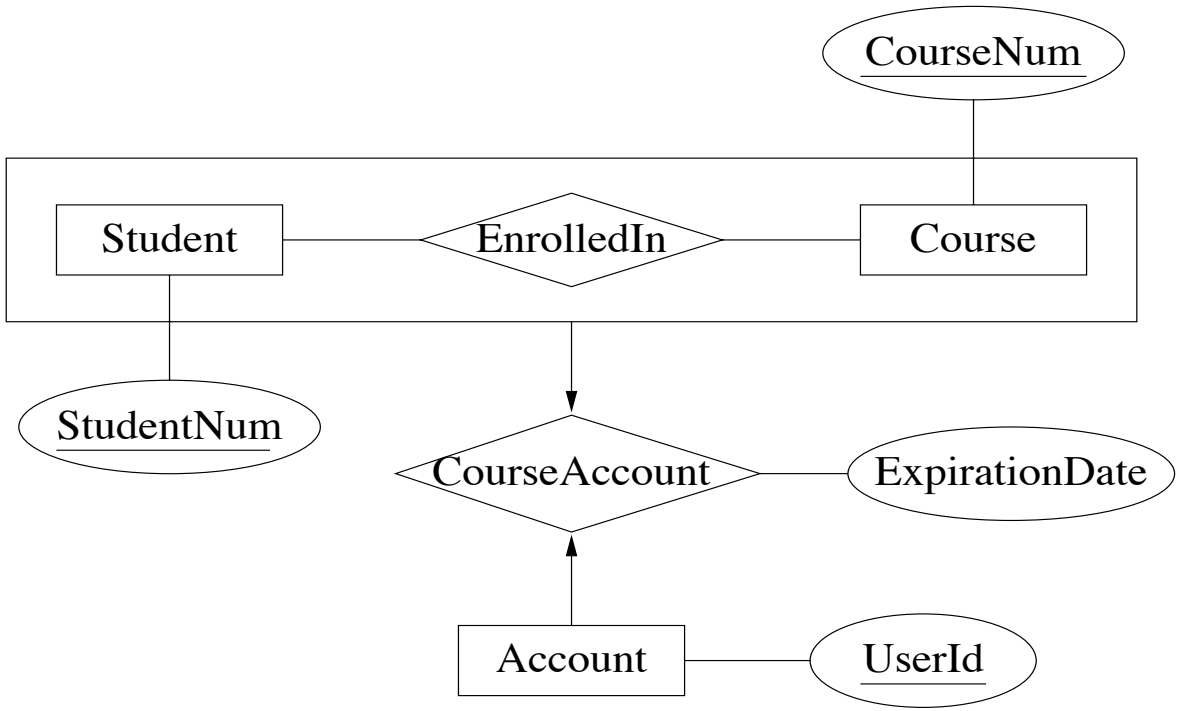
Note: Composite attributes may be multi-valued, and can be a collection of facts about attributes for which some are themselves composite.

Aggregation

Aggregation A relationship set can be aggregated to enable its relationships to be higher-level entities that can in turn participate in other relationships.

Example: *Accounts are assigned to a given student enrollment.*

Graphical visualization:

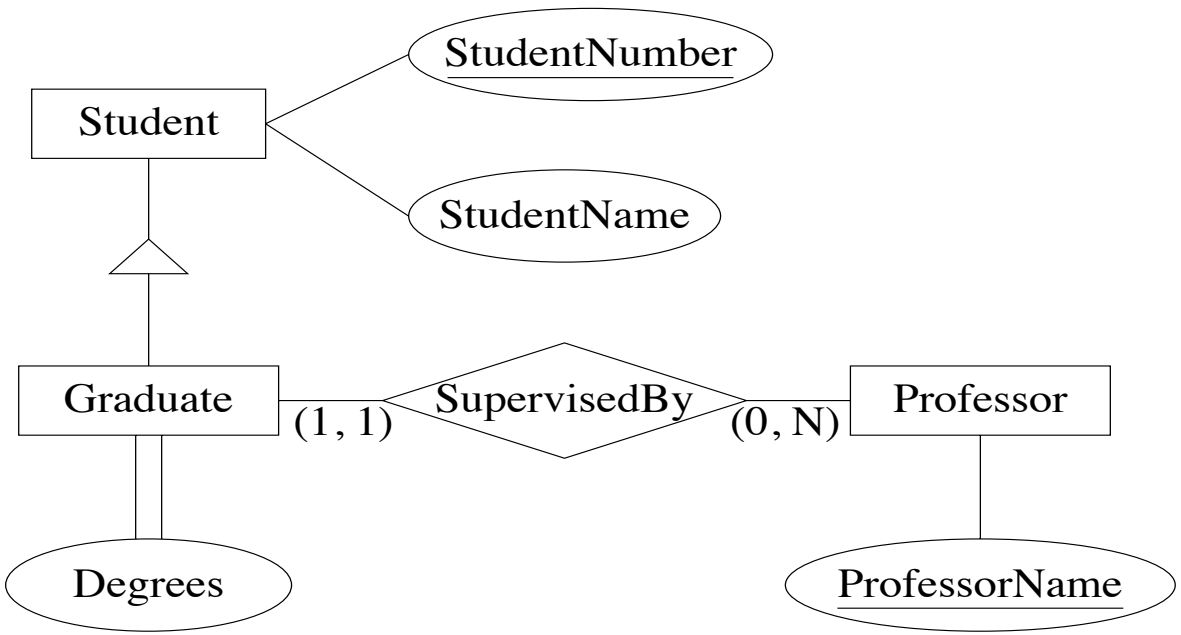


Specialization

Specialization An integrity constraint asserting that the entities of one entity set are also entities of another entity set.

Example: *Graduate students are students who have a supervisor and a number of degrees.*

Graphical visualization:



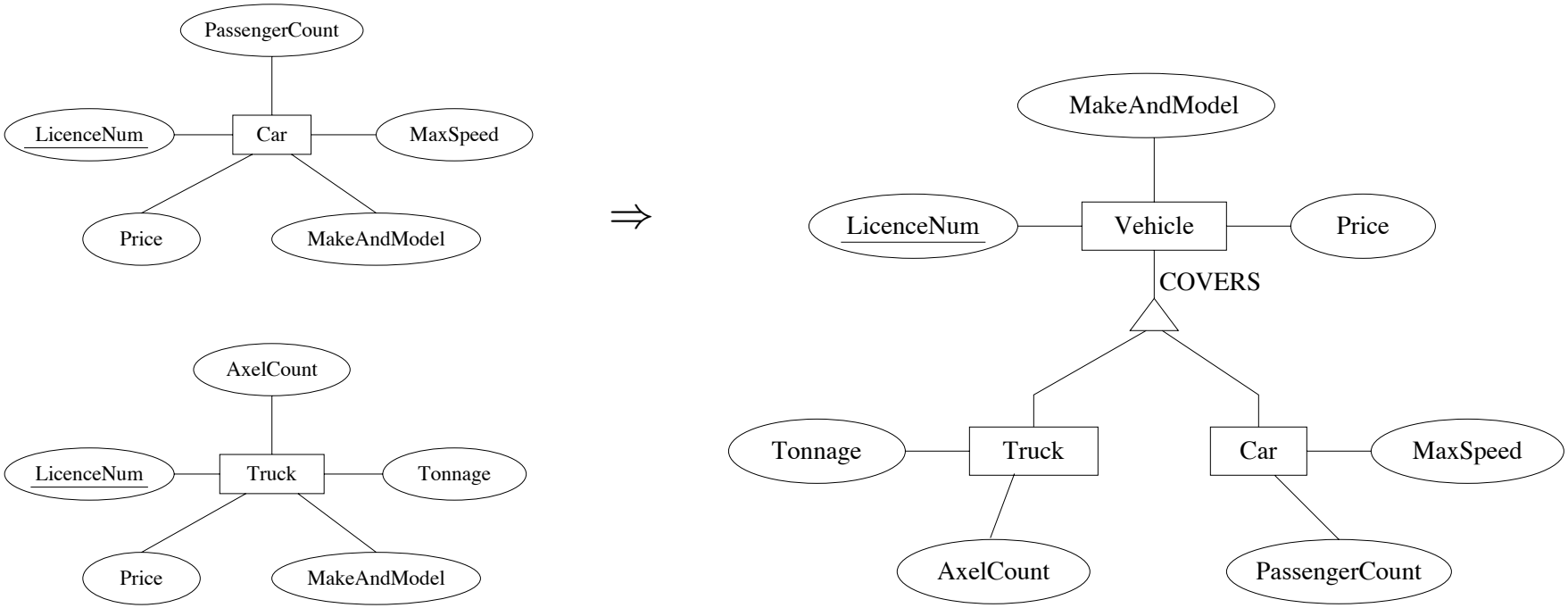
Note: Enables *top down* authoring of an ER diagram.

Generalization

Generalization An integrity constraint asserting that entities of one entity set are also entities of at least one of two or more other entity sets.

Example: *A vehicle is also either a car or a truck.*

Graphical visualization:



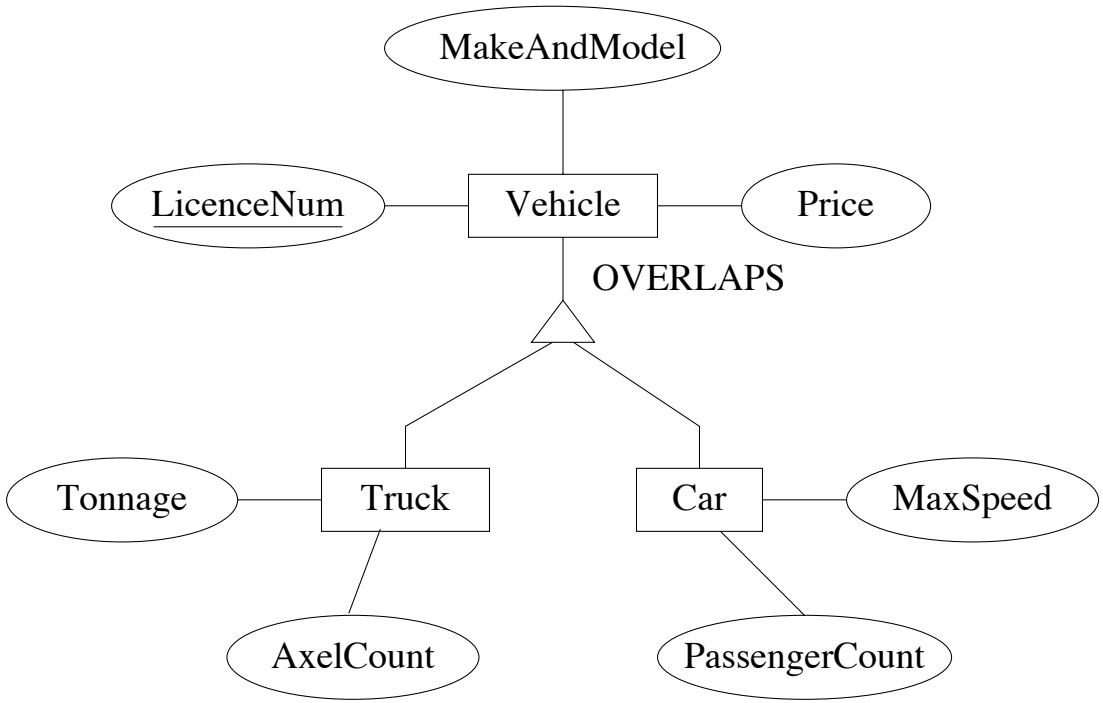
Notes: Enables *bottom up* authoring of ER diagrams. Also, the annotation “COVERS” is optional (and always assumed).

Disjointness

Disjointness Two entity sets participating in a generalization are assumed to be *disjoint* by default. This can be overridden by a graphical annotation on a generalization.

Example: *There are entities that can be both a car and a truck, such as a utility vehicle.*

Graphical annotation:



Semantics via Integrity Constraints and Views

Additional predicates and integrity constraints induced by extended features:

- (*specialization*) *Entity set* `GRADUATE` *is a specialization of entity set* `STUDENT`*.*

  $\forall e.(\texttt{GRADUATE}(e) \rightarrow \texttt{STUDENT}(e))$

- (*generalization and disjunction*) *Entity set* `VEHICLE` *is a generalization of entity sets* `TRUCK` *and* `CAR`*.*

  $\forall e.(\texttt{TRUCK}(e) \rightarrow \texttt{VEHICLE}(e))$

  $\forall e.(\texttt{CAR}(e) \rightarrow \texttt{VEHICLE}(e))$

  $\forall e.(\texttt{VEHICLE}(e) \rightarrow (\texttt{TRUCK}(e) \vee \texttt{CAR}(e)))$

  If generalization is not annotated with “OVERLAPS”, then the entity sets `TRUCK` and `CAR` are disjoint.

  $\forall e.(\texttt{TRUCK}(e) \rightarrow \neg\texttt{CAR}(e))$

- (*aggregation*) `EnrolledIn` *relationships can themselves participate as component entities of* `CourseAccount` *relationships.*

  (see next slide)

Semantics via Integrity Constraints and Views (cont'd)

Use reification on entity set `EnrolledIn`.
Assuming `ENROLLEDIN(student,course)` $\in \rho$, this involves three steps:

1. add to ρ the new predicates

   `ENROLLEDIN-ENT(self)`,

   `STUDENT-COMP(self,student)` and

   `COURSE-COMP(self,course)`

2. add the integrity constraints

   $\forall e.(\texttt{ENROLLEDIN-ENT}(e) \rightarrow \texttt{STUDENT-COMP}(e, -))$,

   $\forall e.(\texttt{ENROLLEDIN-ENT}(e) \rightarrow \texttt{COURSE-COMP}(e, -))$,

   $\forall e_1.(\texttt{STUDENT-COMP}(-, e_1) \rightarrow \texttt{STUDENT}(e_1))$,

   $\forall e, e_1, e_2.(\texttt{STUDENT-COMP}(e, e_1) \wedge \texttt{STUDENT-COMP}(e, e_2) \rightarrow e_1 = e_2)$,

   $\forall e_2.(\texttt{COURSE-COM}(-, e_2) \rightarrow \texttt{COURSE}(e_2))$ and

   $\forall e, e_1, e_2.(\texttt{COURSE-COMP}(e, e_1) \wedge \texttt{COURSE-COMP}(e, e_2) \rightarrow e_1 = e_2)$;

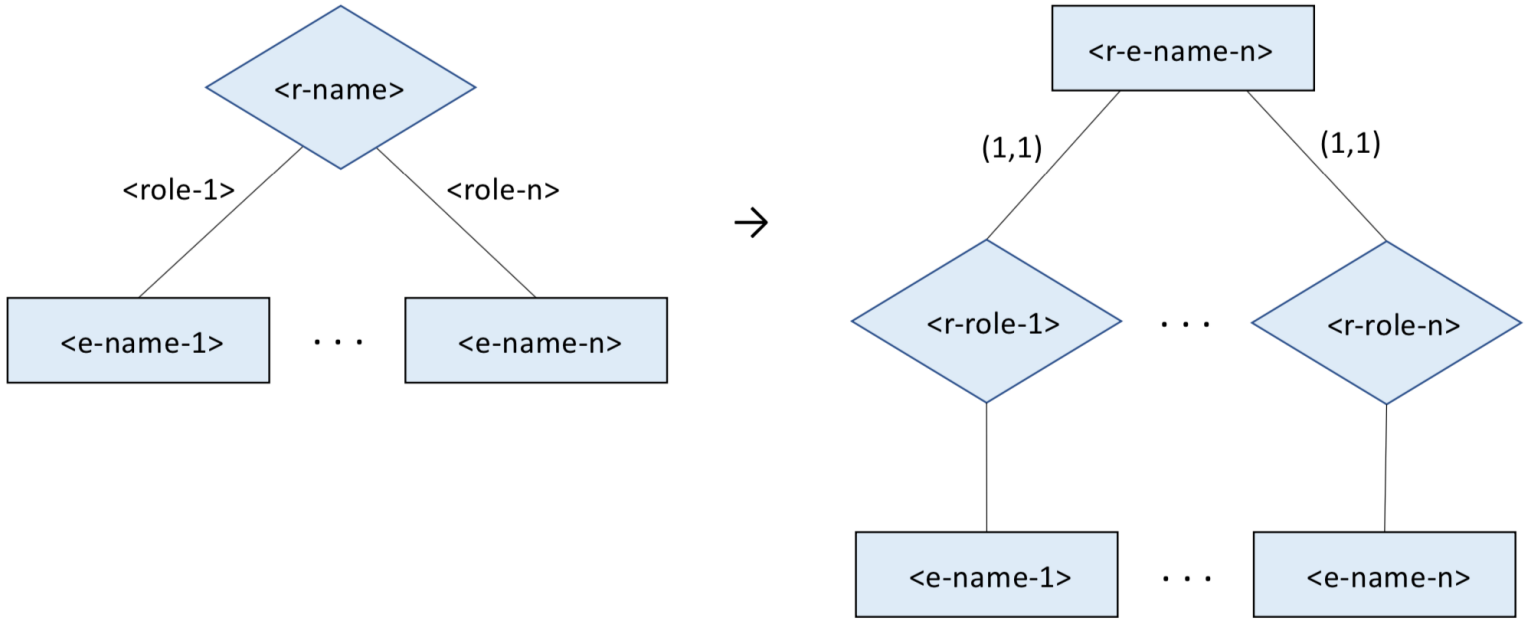
3. and make `ENROLLEDIN` a view with the integrity constraint

   $\forall e_1, e_2.\texttt{ENROLLEDIN}(e_1, e_2)$

   $\leftrightarrow \exists e.(\texttt{ENROLLEDIN-ENT}(e) \wedge \texttt{STUDENT-COMP}(e, e_1) \wedge \texttt{COURSE-COMP}(e, e_2))$.

Semantics via Integrity Constraints and Views (cont'd)

Reification usually replaces a relationship on *n* entity sets with a new entity set with *n* binary relationships on the entity sets.



Aggregation retains both perspectives: viewing an association among *n* entities as an *n*-ary relationship, and as an entity.

Exercise: *Draw an ER diagram for the reification of* `EnrolledIn`.

Outline

Design Methodology

An ER diagram for an information system is usually obtained from two sources:

1. from parts of ER diagrams for existing information systems; and
2. from informal requirements for the information system obtained by *requirements elicitation*.

Issues emerge when authoring an ER diagram from informal requirements.

- When to introduce an attribute versus an entity set.
- When to introduce an entity set versus a relationship set.
- Choosing the arity of relationship sets.
- The use of extended features such as aggregation.
- Methodological considerations.

Attributes versus Entity Sets

Example: *Should one model* `EMPLOYEE` *phones by a* `PhoneNumber` *attribute, or by a* `PHONE` *entity set (with a* `DialNumber` *attribute) that is related to the* `EMPLOYEE` *entity set by a* `HasPhone` *binary relationship set?*

Attributes versus Entity Sets

Example: *Should one model* `EMPLOYEE` *phones by a* `PhoneNumber` *attribute, or by a* `PHONE` *entity set (with a* `DialNumber` *attribute) that is related to the* `EMPLOYEE` *entity set by a* `HasPhone` *binary relationship set?*

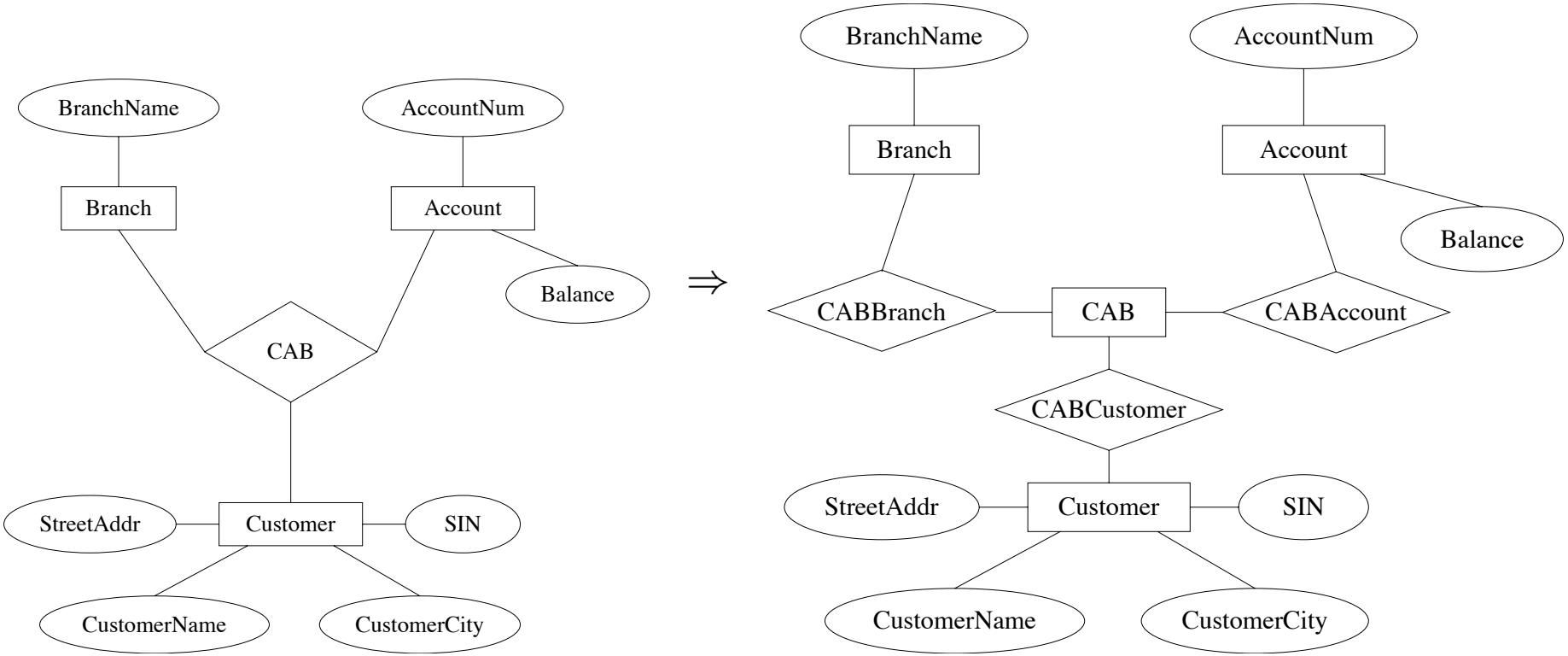
Rules of thumb:

- Is it a separate object?
- Do we maintain information about it?
- Can several of its kind belong to a single entity?
- Does it make sense to delete such an object?
- Can it be missing from some of the entity set's entities?
- Can it be shared by different entities?

An affirmative answer to any of the above suggests going with the `PHONE` entity set.

Entity Sets versus Relationship Sets

Example: *Instead of representing customer branch accounts as ternary relationships, we could represent them as entities.*



Remember that a relationship on *n* entity sets can be replaced with a new entity set with *n* binary relationships on the entity sets via reification.

A Simple Methodology

1. Recognize entity sets.
2. Recognize relationship sets and participating entity sets.
3. Recognize attributes of entity and relationship sets.
4. Define relationship types and existence dependencies.
5. Define general cardinality constraints, keys and discriminators.

For each step, update the ER diagram and maintain a log of assumptions motivating the choices, and of restrictions imposed by the choices.

Case Study: A Registrar's Database

Assume the following informal requirements.

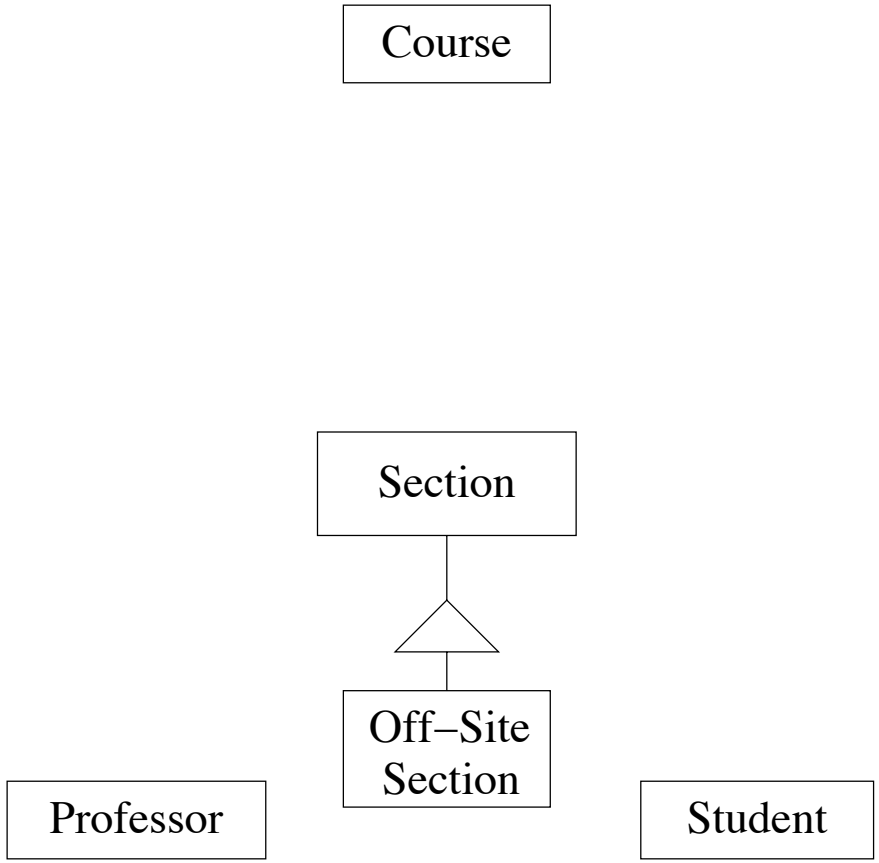
- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)

Step 1. Recognize entity sets,

- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)

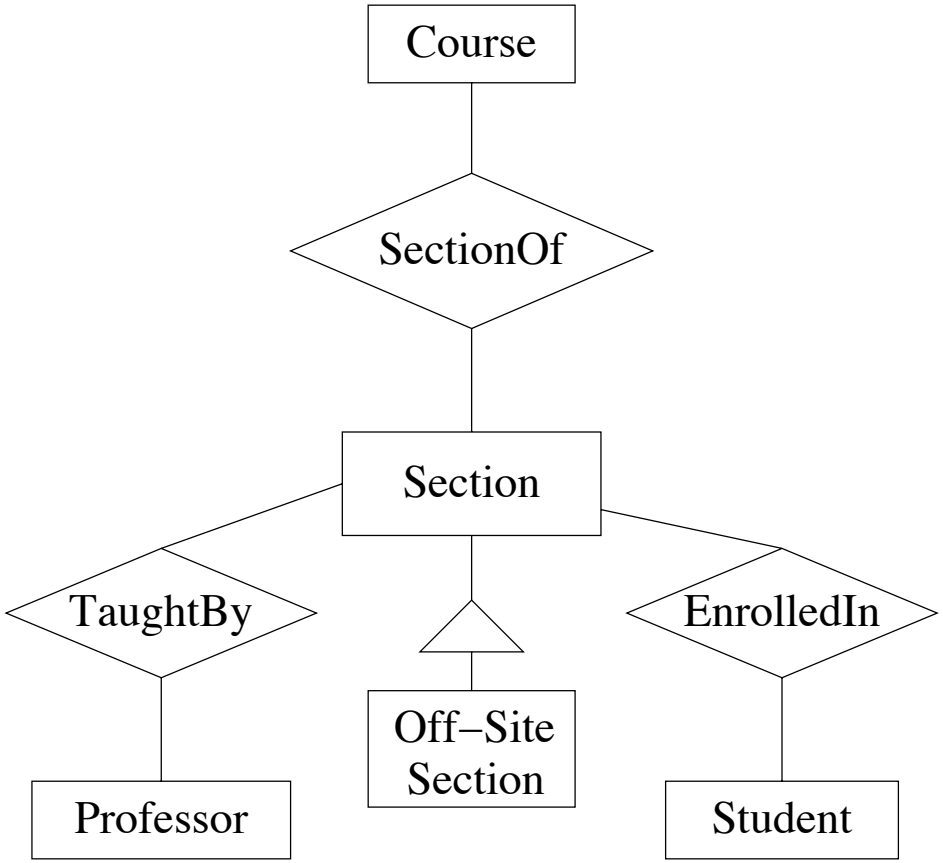


Case Study: A Registrar's Database (cont'd)

Step 2. Recognize relationship sets and participating entity sets.

- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)

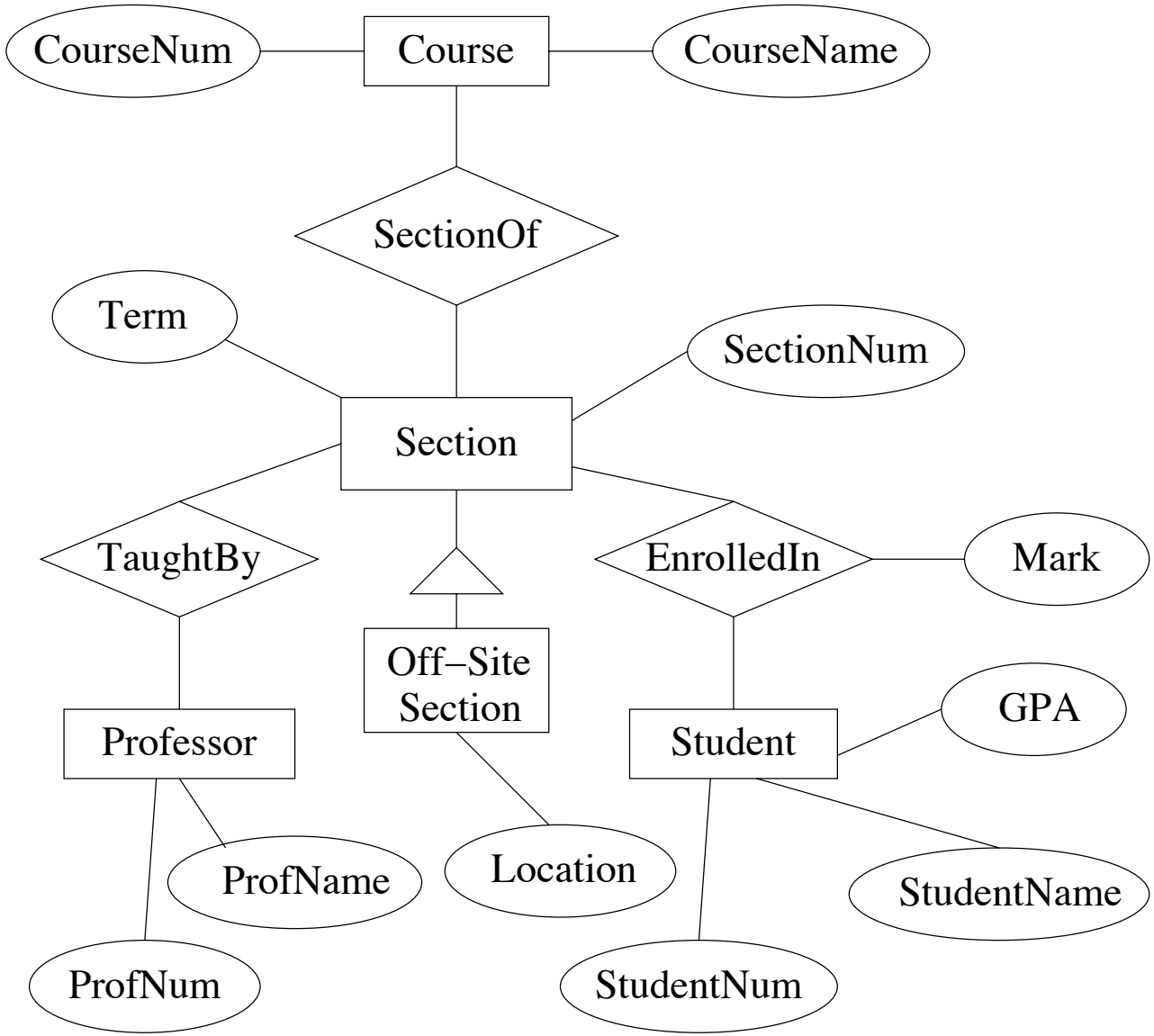


Case Study: A Registrar's Database (cont'd)

Step 3. Recognize attributes of entity and relationship sets.

- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)

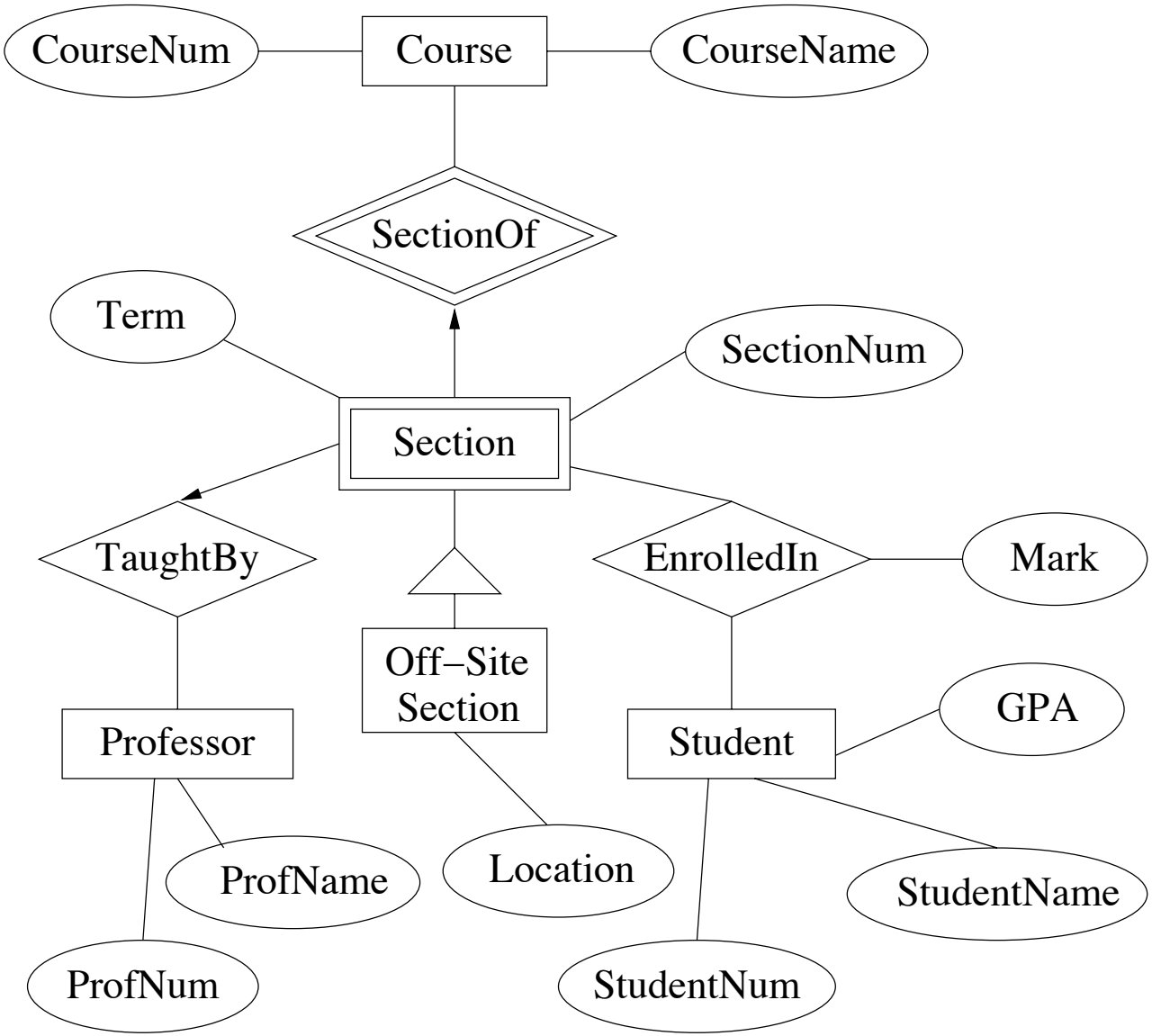


Case Study: A Registrar's Database (cont'd)

Step 4. Define relationship types and existence dependencies.

- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)



Case Study: A Registrar's Database (cont'd)

Step 5. Define general cardinality constraints, keys and discriminators.

- Zero or more sections of a course are offered each term. Courses have names and numbers. In each term, the sections of each course are numbered starting with 1.
- Most course sections are taught on-site, but a few are taught at off-site locations.
- Students have student numbers and names.
- Each course section is taught by a professor. A professor may teach more than one section in a term, but if a professor teaches more than one section in a term, they are always sections of the same course. Some professors do not teach every term.
- Up to 50 students may be registered for a course section. Sections with 5 or fewer students are cancelled.
- A student receives a mark for each course in which they are enrolled. Each student has a cumulative grade point average (GPA) which is calculated from all course marks the student has received.

Case Study: A Registrar's Database (cont'd)