Information Systems

WS 2005, JKU Linz
Course 2: The Entity-Relationship (ER) Model

Gábor Bodnár

URL: `http://www.risc.uni-linz.ac.at/education/courses/ws2005/is/`

Overview

- Entities, Attributes, Relationships.
- Keys.
- ER diagrams, ER design.

The Entity-Relationship Model

It was created by P. Chen in 1976 together with the ER diagramming technique.

The purpose of semantic modeling is

- to attempt to represent meaning of data in a database,
- to support flexible design.

ER modeling is a semantic modeling technique that views the real world as sets of *entities*, described by *attributes*, which have *relationships* between each other.

Entities

An *entity* is an object with identity, i.e. it is distinguishable from other objects.

Entities can be concrete or abstract objects: persons, documents, events, programs, etc.

An *entity set* is the set of all entities which we consider to be of the same type from our viewpoint.

Analogies exists between entity sets and variable types, and entities and variables.

For instance, the set of students of the JKU can be considered as an entity set.

Attributes

An entity is described by *attributes*.

An *attribute* is a function from an entity set to some *domain* of permitted values.

For instance, let S be the entity set of students of the JKU, and let A^+ be the set of nonempty character sequences of the Latin alphabet. Then an example of an attribute can be

$$\mathrm{Name} : S \rightarrow A^+ \times A^+.$$

Remark: as we can see, attributes can have composite values; sometimes even set-valued functions are allowed.

Relationships

A *relationship* is just an association among entities.

A *relationship set* is the set of relationships of the same type.

Using the usual mathematical definition a relationship set over $E_1, \ldots, E_n$ is a relation over $E_1, \ldots, E_n$, that is, a subset of $E_1 \times \cdots \times E_n$.

By an abuse of terminology, we often say relationship instead of relationship set.

A relationship can be considered as an abstract entity, thus a relationship set can be represented just as an entity set.

The *degree* of a relationship (set) is the number of entity sets associated via it.

Example

Entity set: `Student`.

Attributes:

- `fname`: `Student` $\longrightarrow A^+$,
- `lname`: `Student` $\longrightarrow A^+$,
- `sid`: `Student` $\longrightarrow \{$'0','1','2','3','4','5','6','7','8','9'$\}^7$.

An entity (instance) is described via its attributes. For example the entity X is identified by:

$\{$(`fname`$(X)=$ `'Susan'`), `lname`$(X)=$`'Becker'`, `sid`$(X)=$`'0256098'`$\}$.

Example (continued)

Entity set: `Course`.

Attributes (using now a simplified, “computer science” notation):

- `title`: `char`(40),
- `cid`: ['0'-'9'](6).

Then the relationship which expresses that Susan Becker takes the Analysis I course can be described, with a naive approach which uses all the attributes, by the entity Y (that represents the relationship):

(`fname, lname, sid, title, cid`) $=$
(`'Susan','Becker','0256098','Analysis I','322527'`).

Classification of Relationships

Higher degree relationships can always be resolved into binary ones by introducing abstract entities (an example comes later).

The *(mapping) cardinality* of a relationship describes the number of entities that can be associated with each other via the relationship.

A binary relationship between entity sets A and B can be

One-to-One (1:1), An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

Example: The relationship between students and ID cards.

Classification of Relationships

One-to-Many (1:N) An entity in A is associated with any number of entities in B. An entity in B is associated with at most one entity in A.

Example: The relationship between members and instances of books of a library.

Many-to-Many Entities in A and B are associated with any number from each other.

Example: The relationship between students and courses at a university.

Classification of Relationships

The *existence dependency* of a binary relationship expresses that the existence of an entity is dependent on another, related entity.

If in a relationship between entities X and Y, the existence of X depends on the existence of Y, we call Y the *dominant* and X the *subordinate* entity.

Example: In the relationship of students and IDs, we can declare the student entity set as the dominant and the ID as the subordinate.

The *direction* of a relationship (if given) is from the originating entity set (called parent) to the other (called child).

In a 1:1 relationship the dominant entity set is the parent.

In a 1:N relationship the entity set on the '1'-side is the parent.

Keys

Since entities are described by attributes and entities must be distinguishable, there should always be a subset of attributes whose values make the distinction possible.

A *superkey* a nonempty set of attributes which allows us to uniquely identify an entity of the entity set.

A superkey which does not have a proper subset which is also a superkey is called a *candidate key*.

A *primary key* is a candidate key which is chosen by the database administrator to identify entities in an entity set.

Remarks: Usually there are many superkeys and more than one candidate keys.

Example

Take again the entity set: `Course`, with attributes

- `title`: `char`(40),
- `cid`: ['0'-'9'](6).

Then we have that

- {`title`,`cid`} is a superkey,
- {`cid`} should clearly be a candidate key, and it is also a good choice to be a primary key,
- if there are no two identical course names, {`title`} is also a candidate key.

Keys (continued)

An entity set that does not possess sufficient attributes to form a primary key is called a *weak entity set*.

A *strong entity set* on the other hand has a primary key.

A weak entity set must be in a 1:N relation with a strong entity set on which it must existence depend.

A *discriminator* of a weak entity set is a set of attributes which allows the distinction of the entities when taken together with the primary key of the strong entity set on which it existence depends.

A primary key for a weak entity set is then formed by the primary key of the related strong entity set plus its discriminator.

Example

Weak entity set: `APhone`, attribute: `ph-nr` (['0'-'9'](7)).
Local phone numbers in Austria.

Strong entity set: `ACodes`, attribute: `code` (['0'-'9'](3)).
Phone area codes in Austria.

The 1:N relationship between `ACodes` and `APhone` associates the phones to the area they belong.

The entities of `APhone` existence depend on the ones of `ACodes`.

The discriminator of `APhone` is `ph-nr` and the primary key is {`code`, `ph-nr`}.

Keys (continued)

The primary key of a relationship set is formed from the primary keys of the entity sets in the relationship set.

A *foreign key* is a set of attributes for an entity set in a 1:N binary relation on the "N" side, which identifies the corresponding parent entity for each entity of the set.

Foreign keys provide means to maintain data integrity in databases (called referential integrity).

Example

- Entity set: `Member`, attributes: <u>`mid`</u>, `name`.
- Entity set: `Book`, attributes: <u>`id`</u>, `isbn`, `mid`.

Example (continued)

The 1:N relationship `Borrows` associates each book to its borrower.

The primary key of the relationship is `mid, id`
(to avoid ambiguity we can write `Member.mid, Book.id`).

The `Book.mid` attribute can contain:

- NULL, if the book is in the library,
- the ID of the member who borrows it, otherwise.

Thus {`Book.mid`} is a foreign key in `Book` for `Member` verifying the relation `Borrows`.

ER Diagrams

ER diagram is a graphical way to express the logical structure of a database.

There are many “dialects” of ER diagrams; here is one:

- Entity sets are represented by labeled rectangles, for weak entity sets the rectangles have double border. Entity set labels (names) should be singular nouns.

- Relationship sets are represented by a solid line connecting two entity sets. The name of the relationship is written beside the line. Relationship names should be verbs.

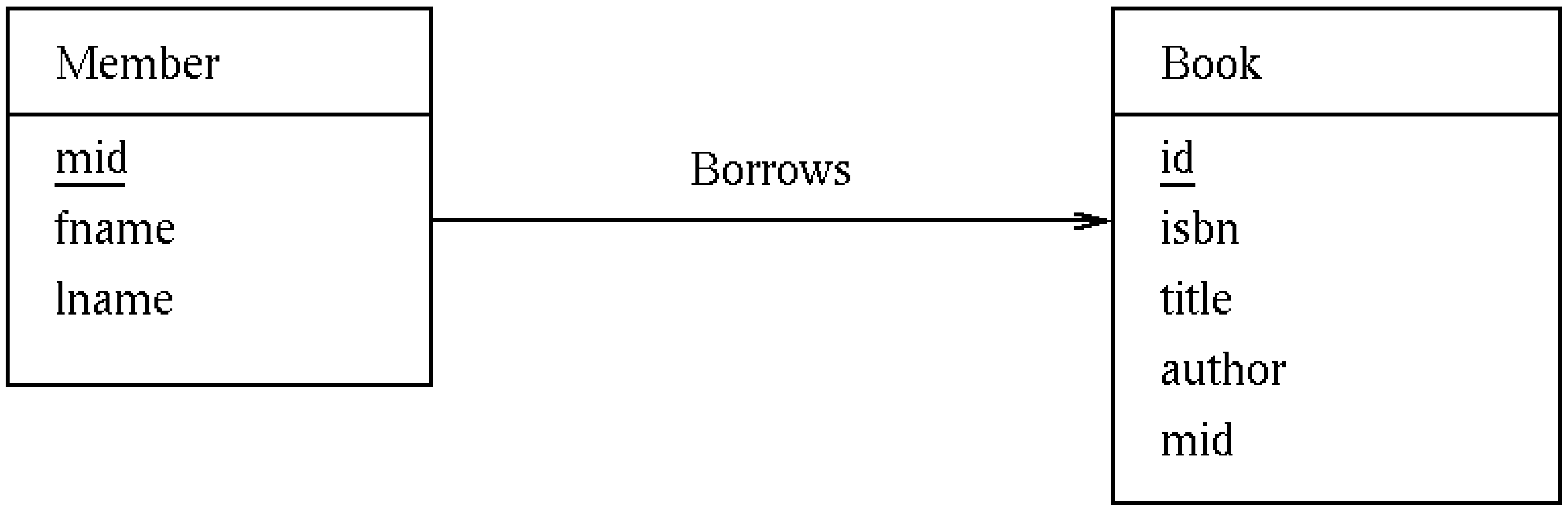
ER Diagrams (continued)

- Attributes, when included, are listed inside the entity set rectangle. Attribute names should be singular nouns. Members of the primary key can be denoted by underlining.
- If the line ends in an arrow, the relationship cardinality is Many for the entity set at that end. Otherwise the cardinality is One.
- Existence dependency is denoted by a bar crossing the line of the relationship set at the dominant entity set.

An ER diagrams can describe the logical structure of a database in an intuitive format which is easy to work with.

Example

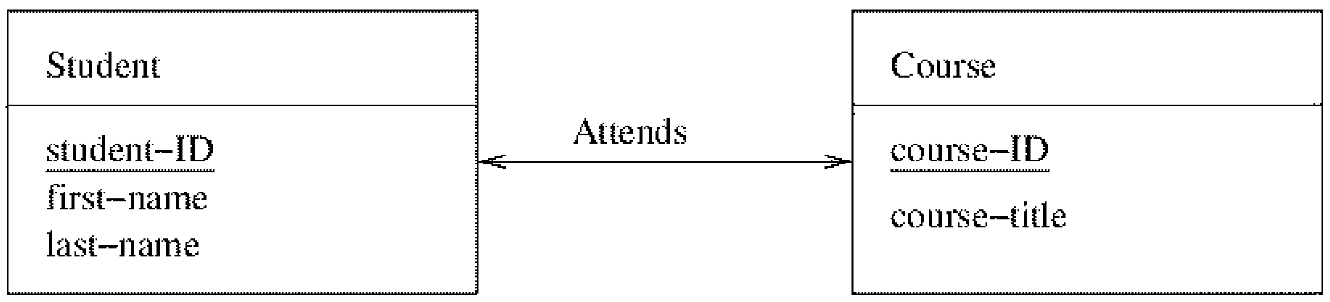
An extended version of the entity sets `Member`, `Book` and the relationship `Borrows`.

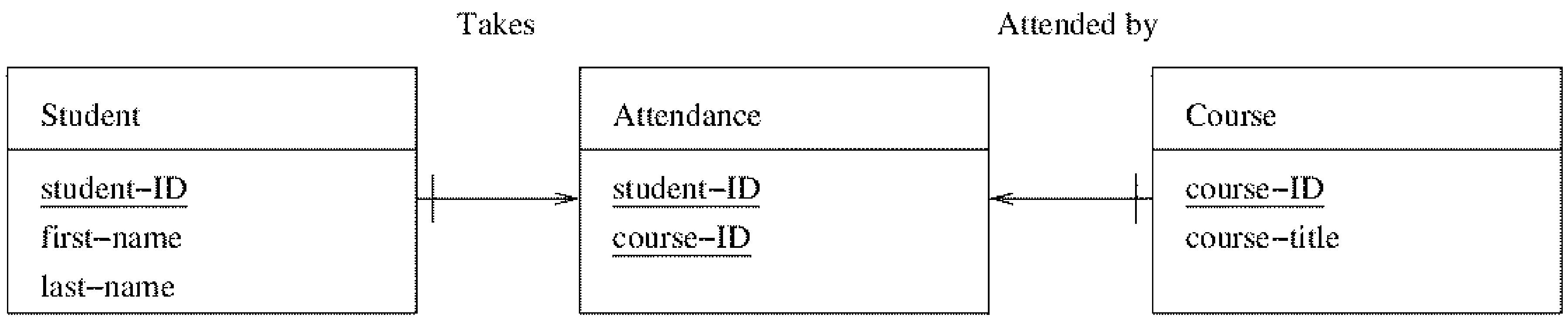


A database conforming to an ER diagram can be represented by a set of tables, each corresponding to an entity set or to a relation.

Resolving Complex Relationships

However, M:N relationships cannot be taken over directly to the relational model; therefore they have to be resolved into 1:N relationships by introducing new entity sets.





Resolving Complex Relationships

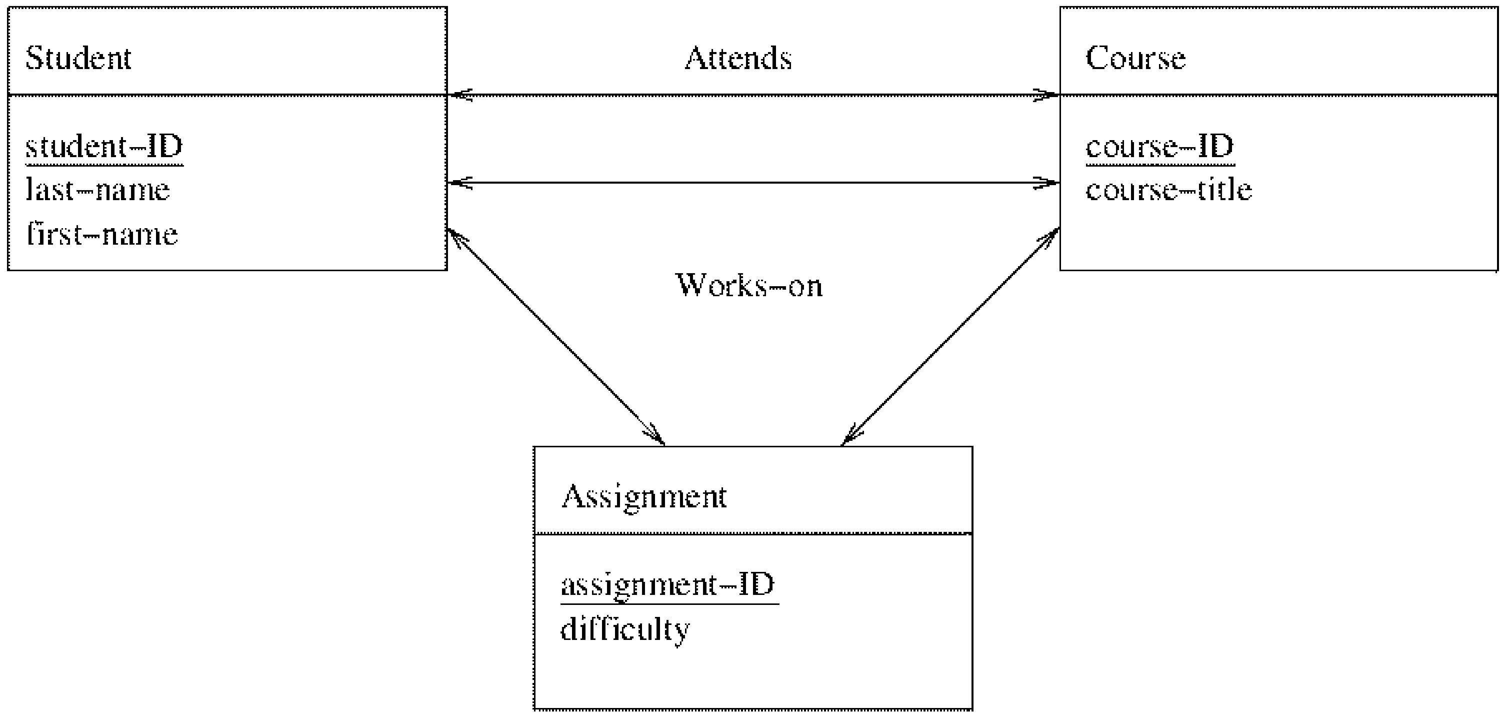
Relationships of degree greater than two should be resolved into binary 1:N relationships using the technique of the previous slide.

The introduction of new entity sets for relationships into the model and establishing relationships between such “relationship-entity sets” is called *aggregation*.

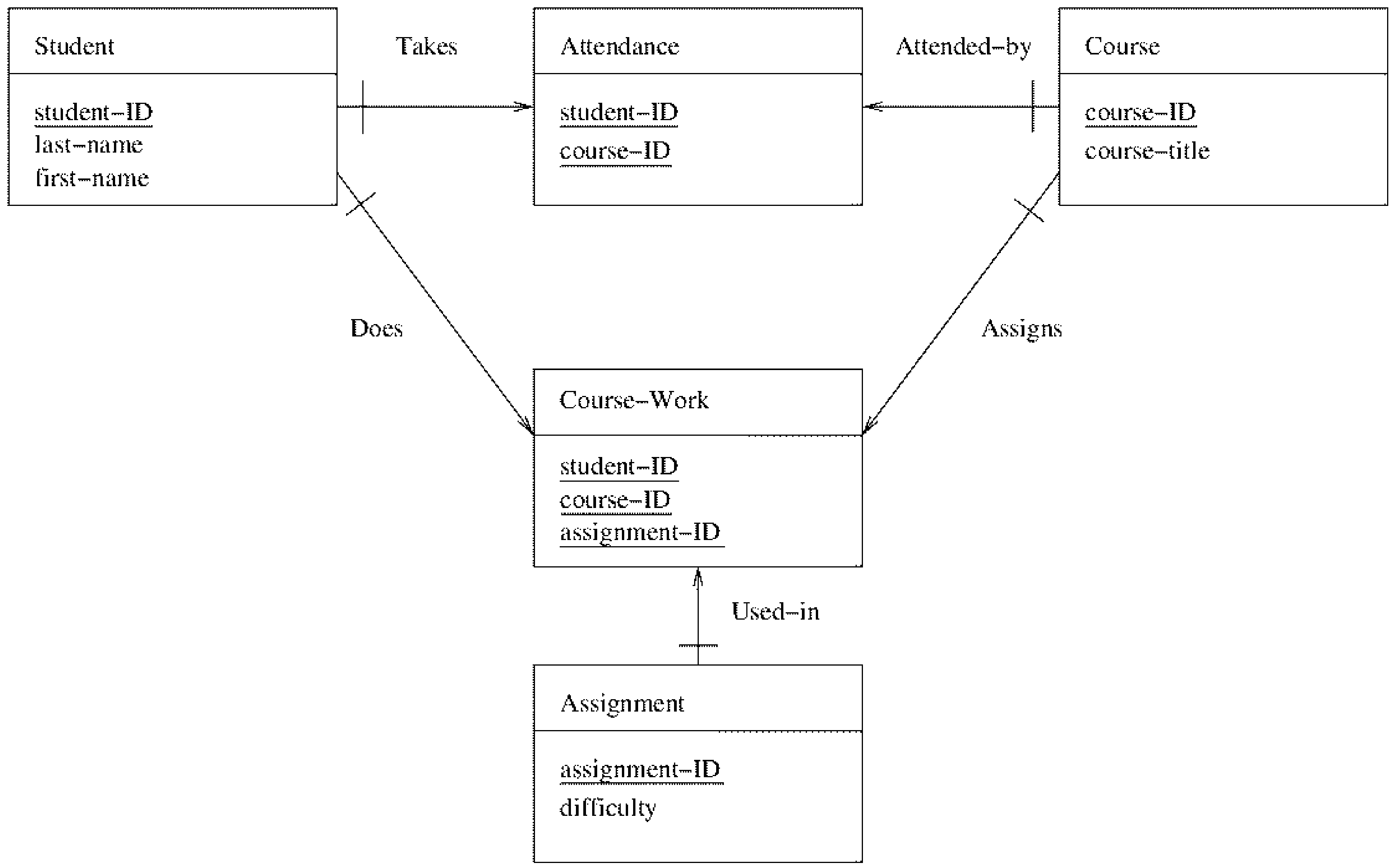
Generalization, on the other hand, introduces new entity sets which collect certain common properties of other entity sets of the model.

The newly created entity set is also referred to as *super type*, while its specializations as *subtypes*.

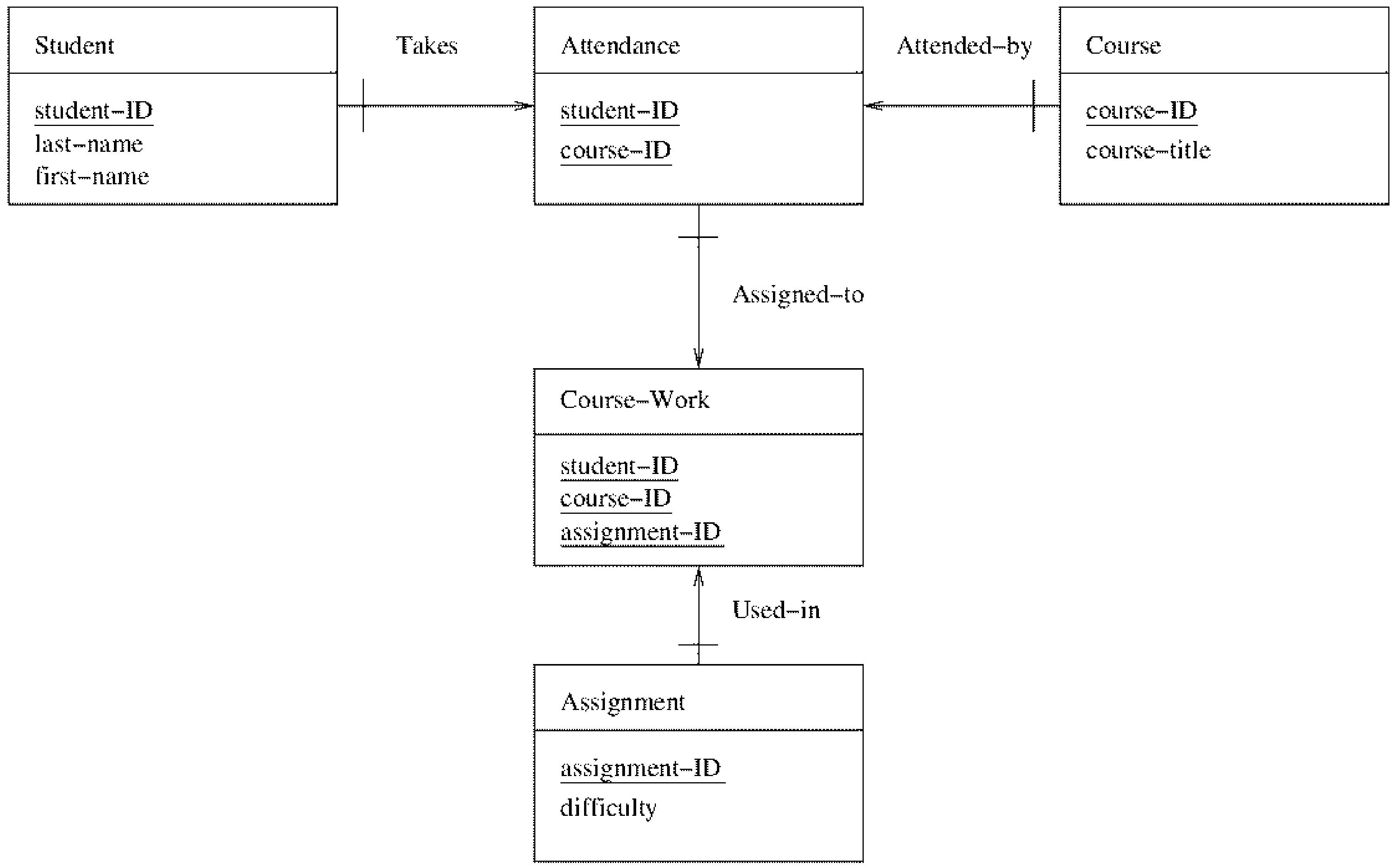
Example


Student
student–ID
last–name
first–name
Attends
Course
course–ID
course–title
Works–on
Assignment
assignment–ID
difficulty

Example


Student
student–ID
last–name
first–name
Takes
Attendance
student–ID
course–ID
Attended–by
Course
course–ID
course–title
Does
Assigns
Course–Work
student–ID
course–ID
assignment–ID
Used–in
Assignment
assignment–ID
difficulty

Example


Student
student–ID
last–name
first–name
Takes
Attendance
student–ID
course–ID
Attended–by
Course
course–ID
course–title
Assigned–to
Course–Work
student–ID
course–ID
assignment–ID
Used–in
Assignment
assignment–ID
difficulty

ER Design

Goals of requirement analysis:

- Determine what data (viewed as elementary objects) is necessary to be included in the database.
- Collect the descriptive information about these objects.
- Identify and classify relationships between these objects.
- *Identify what transactions will be executed on the database and how will they affect the data objects.*
- *Identify data integrity rules.*

ER Design

Data modeling:

- Identification of entity and relationship sets.
- Drafting the initial ER diagram.
- Refining the ER diagram by resolving complex relationships.
- Add key attributes to the diagram.
- Adding non-key attributes.
- Constructing generalization hierarchies.
- Adding integrity rules to the model.

Summary

- Entities, Attributes, Relationships
- Degree, Mapping cardinality of relationships
- Existence dependency
- Superkeys, Candidate keys, Primary keys
- Weak and Strong entity sets
- ER diagrams, Resolution of Complex Relationships
- ER design recommendations