Data Models

Week 2 - SIT103
Entity Relationship Diagrams

Relational Model

Models

- A model in general is an ‘abstract’ of reality.
- What are some examples of models?

Scale model of a building

Scale model of a plane

- These can be created using cardboard or CAD/CAM software or clay models.
- We can build model things which exist nor or does not exist yet.

Models

- Models can be expressed in different ways using different tools.
- Used to show same thing in different ways
- Show different levels of detail
- Target different audiences
- Remove irrelevant detail.
- Example
  - scale clay model of a car showing how it would look
  - blueprints of the car, showing all measurements etc.
  - computer simulation of the car in working order

What are Models used for?

- The general purpose for models are:
  - getting a better “overall” view of a real life application;
  - communication between same level;
  - communication to different levels.
- In databases, we can model an actual database and/or reality.

Data Models

- Entity Relationship Models
- Relational Model
- Object Orientated Models (UML)

NOTE: In the end all these models will model the same reality/end product.

IMPORTANT: Only values are not displayed in data models

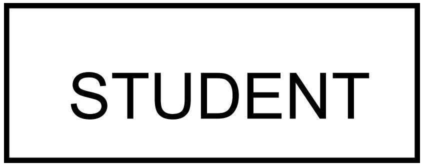
Entity Relationship Diagrams

- Most popular model currently used for database modelling
- Designed to be platform independent
- Primary components
  - **Entities**
  - **Relationships**
  - **Attributes**

ENTITY

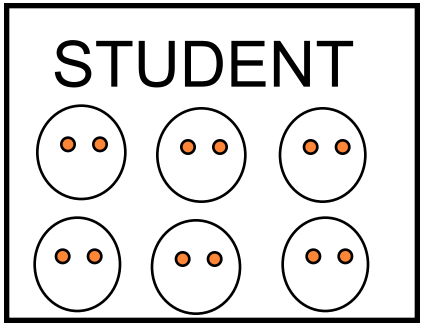
- Object or Event. Real or Imaginary
- Interesting enough to us that we want to record something about it
- Entity drawn as a box and a name
- Represents a collection of individual objects that we can uniquely identify from each other.
- Objects are called “entity instances”

ENTITY



STUDENT

- Same thing as a table
- This box is the Entity STUDENT
- The entity actually represents the collection of individual students

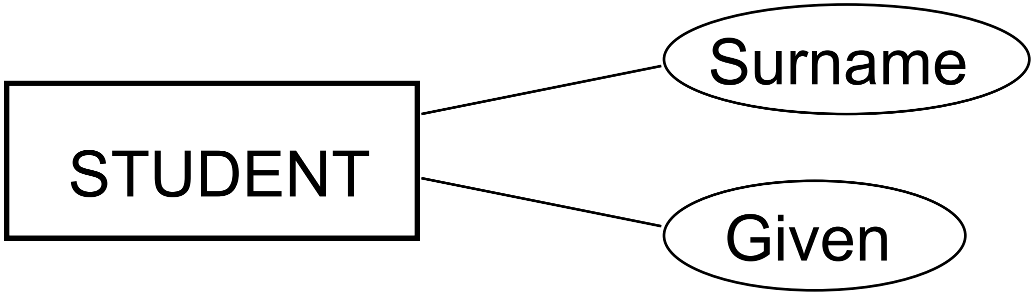


WEAK ENTITY

- **Weak entity** : An entity that does not have a primary key.
  - its existence depends on the existence of one or more *identifying entity*
  - it must relate to the identifying entity via a total, one-to - many identifying relationship set from the identifying to the weak entity set
  - Examples:
    - ***child*** from parents, ***payment*** of a loan
- The **partial key** of a weak entity is the attribute that distinguishes among all the entities of a weak entity related to the same owning entity.

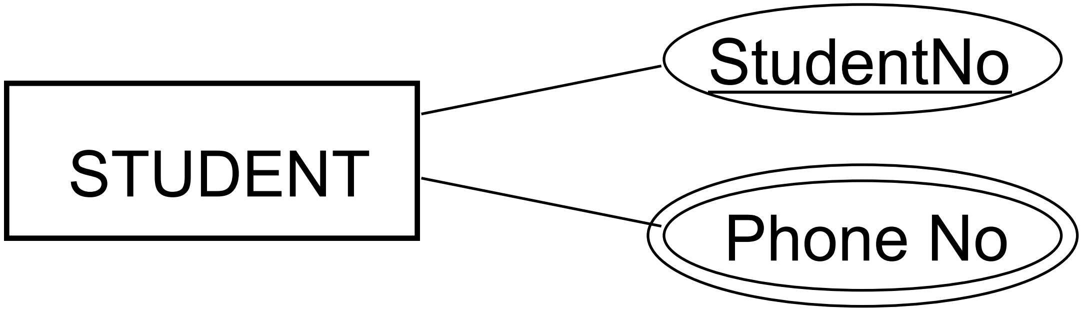
ATTRIBUTE

- Same thing as a field or column
- A property or characteristic of an entity
- A piece of data that we want to keep about the entity.
- Drawn as an ellipse connected to the entity.



Types of Attributes

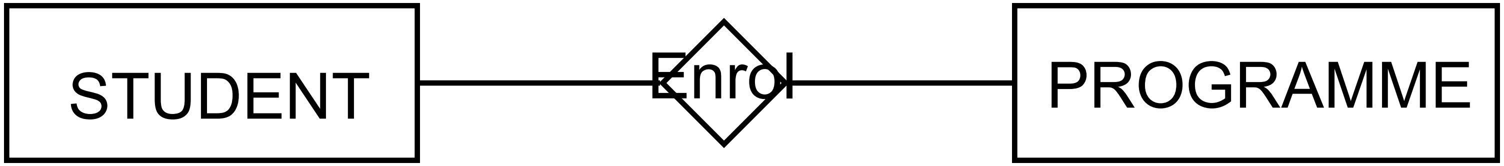
- The primary key attribute is underlined.
- A multi-valued attribute is an attribute that can have several values. It is shown in a double ellipse.
  - Multi-valued attribute example: Phone
  - Phone can have:
    - Country Code
    - Area Code
    - It can be mobile or a land line



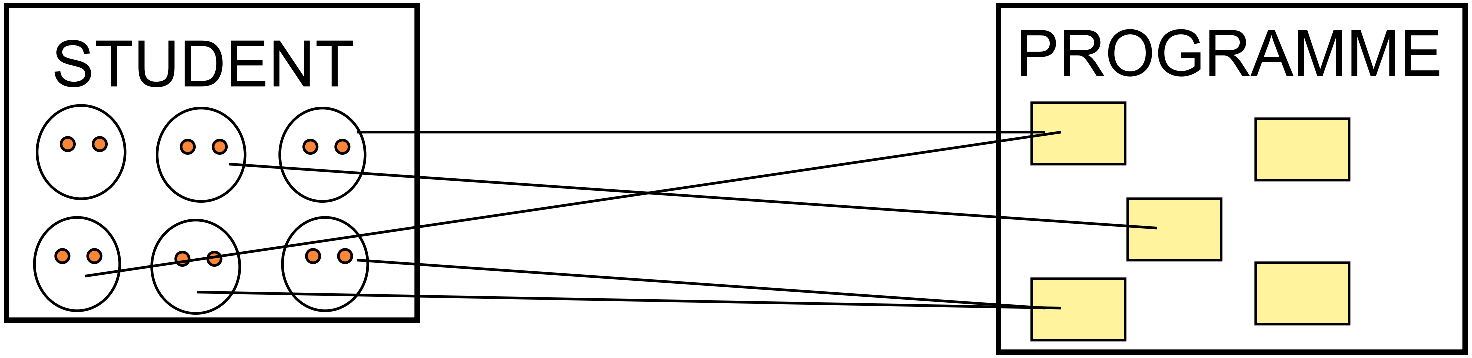
RELATIONSHIP

- Some type of association between entities
- Shown as a diamond joining entities together with a relationship name.
- Therefore, the Relationship represents the collection of individual associations.
- We could use relationships for join

RELATIONSHIP



- This diamond represents the collection of individual associations between individual students and programmes

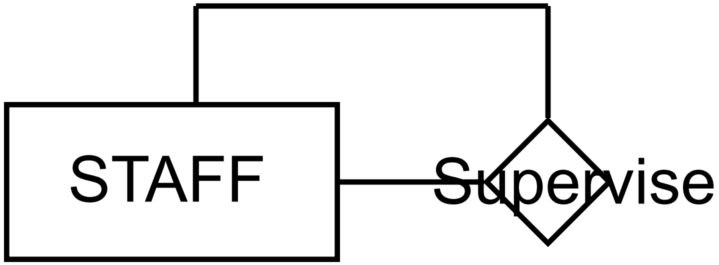


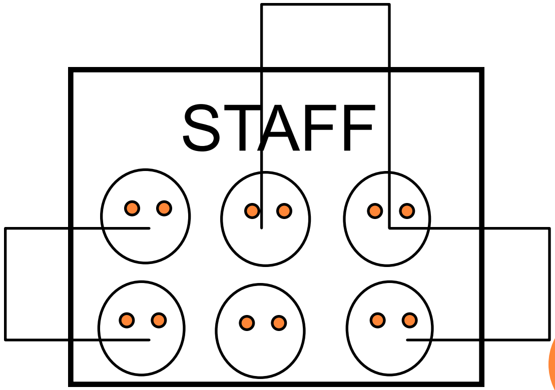
DEGREE

- Number of Entities a relationship is involved in
  - Unary (Only one entity)
  - Binary (Two entities)
  - Ternary (Three entities)
  - N-ARY (More than two entities)

DEGREE (UNARY) – ONE ENTITY

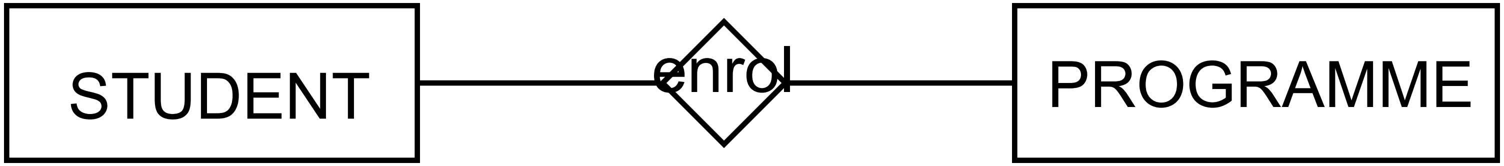
- A Unary relationship is a relationship of an entity to itself.
- The associations are between different objects of the same entity.
- Designed for displaying a **self-join**





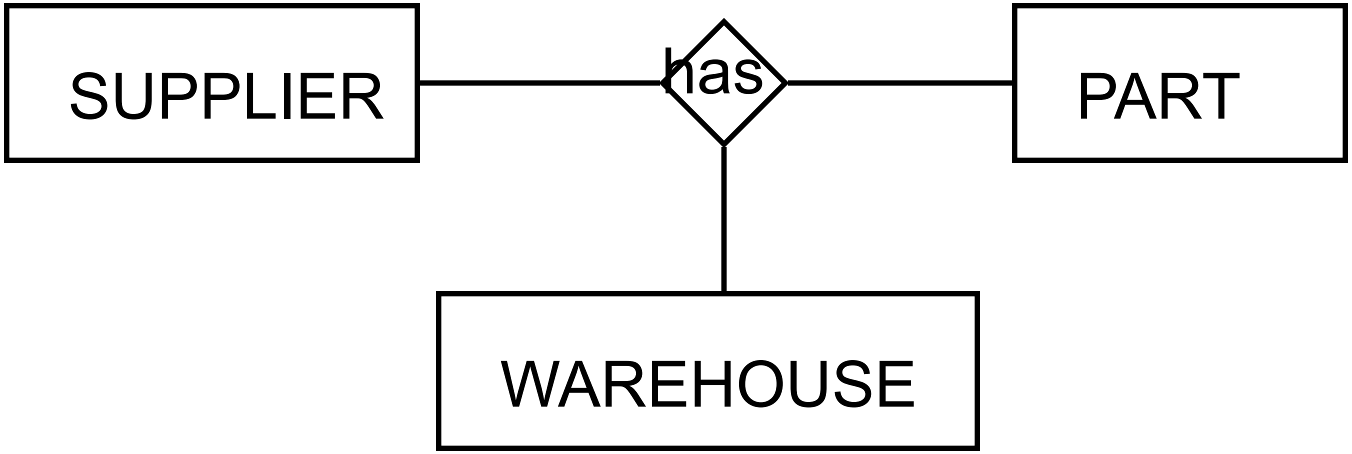
DEGREE (BINARY) – TWO ENTITY

- Relationship between two entities
- Most common
- Designed for displaying a natural join between 2 tables



DEGREE (TERNARY & N-ARY)

- Ternary is between 3 entities
- We can actually have a relationship between any number of entities.

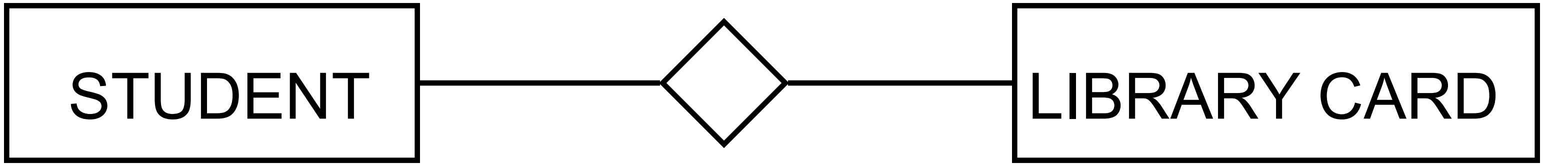


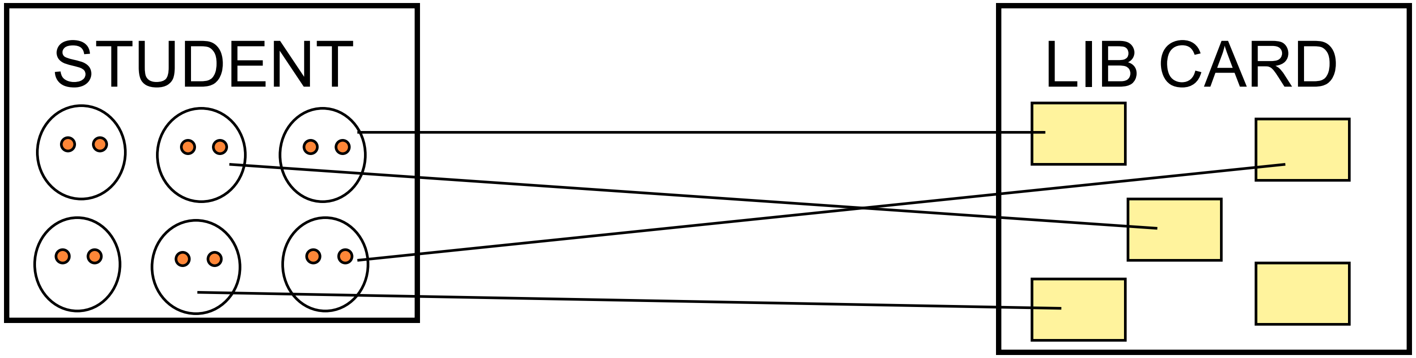
Cardinality

- Determines how many associations are allowed between individual objects involved within a relationship.
- Shown using "crow's feet" , using an arrow head (refer tutorial for more details) or with 1:M
- Enables us to constrain the relationships to only allow particular situations (how many courses can a student have ?)

Cardinality (One to One)

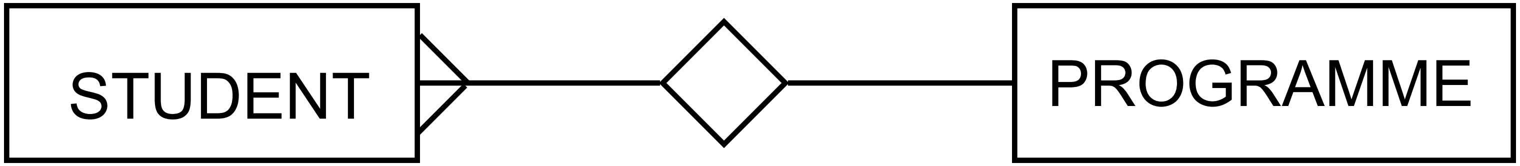
- A student can only have 1 library card
- A library card can only be for 1 student

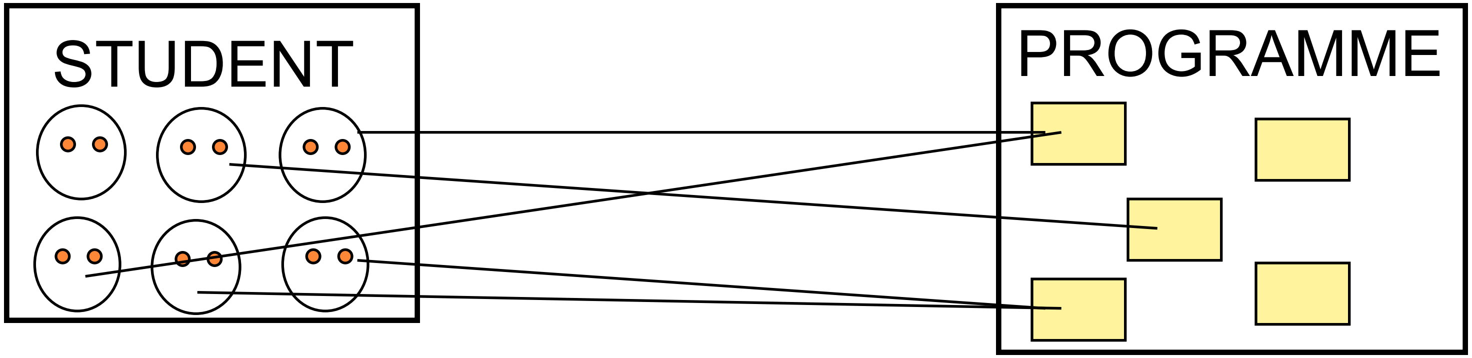




CARDINALITY (ONE TO MANY)

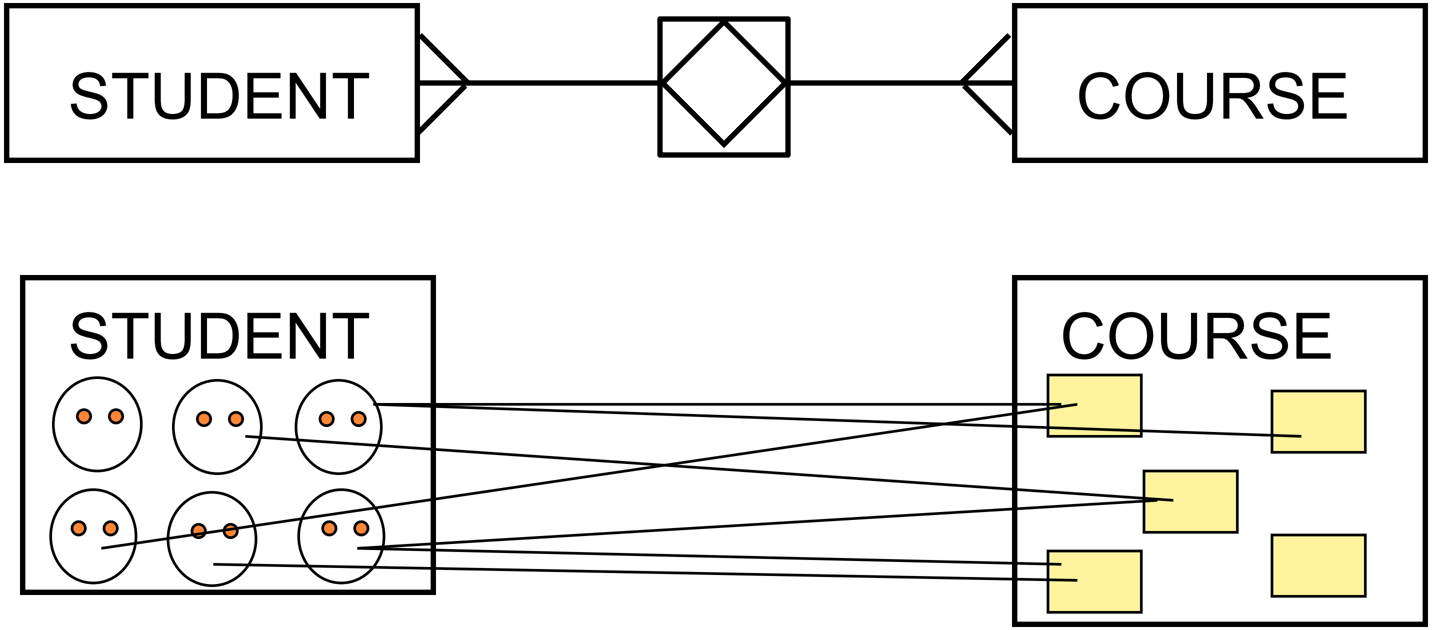
- A student can only have one programme
- A programme can have many students





Cardinality (Many to Many)

- A student can have many courses
- A course can have many students

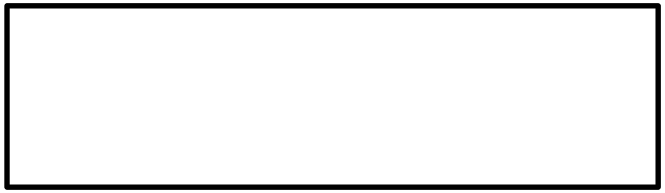


Relationships

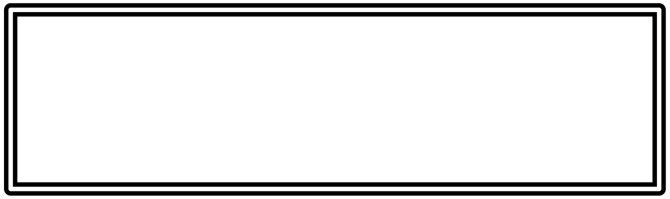
- Relationships can also have attributes
- There is no clear rule to determine between a relationship and an entity
- Often, an entity can become a relationship and a relationship can be re-drawn as an entity.

Crow’s Foot Notation -Relationships

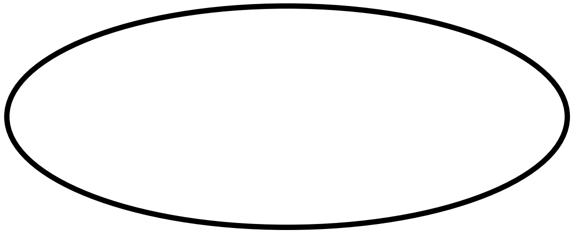
- Entity



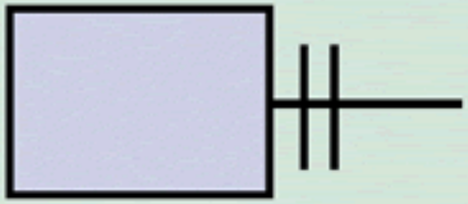
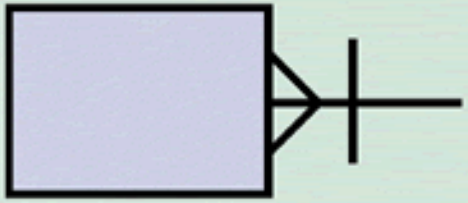
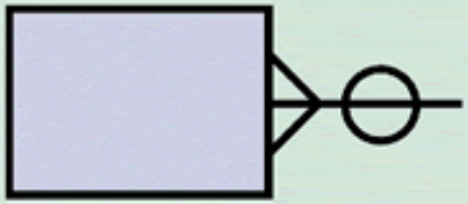
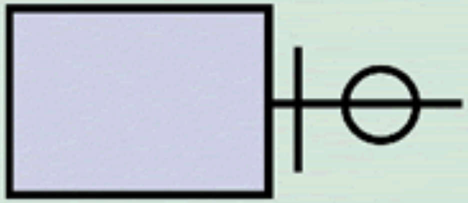
- Weak Entity



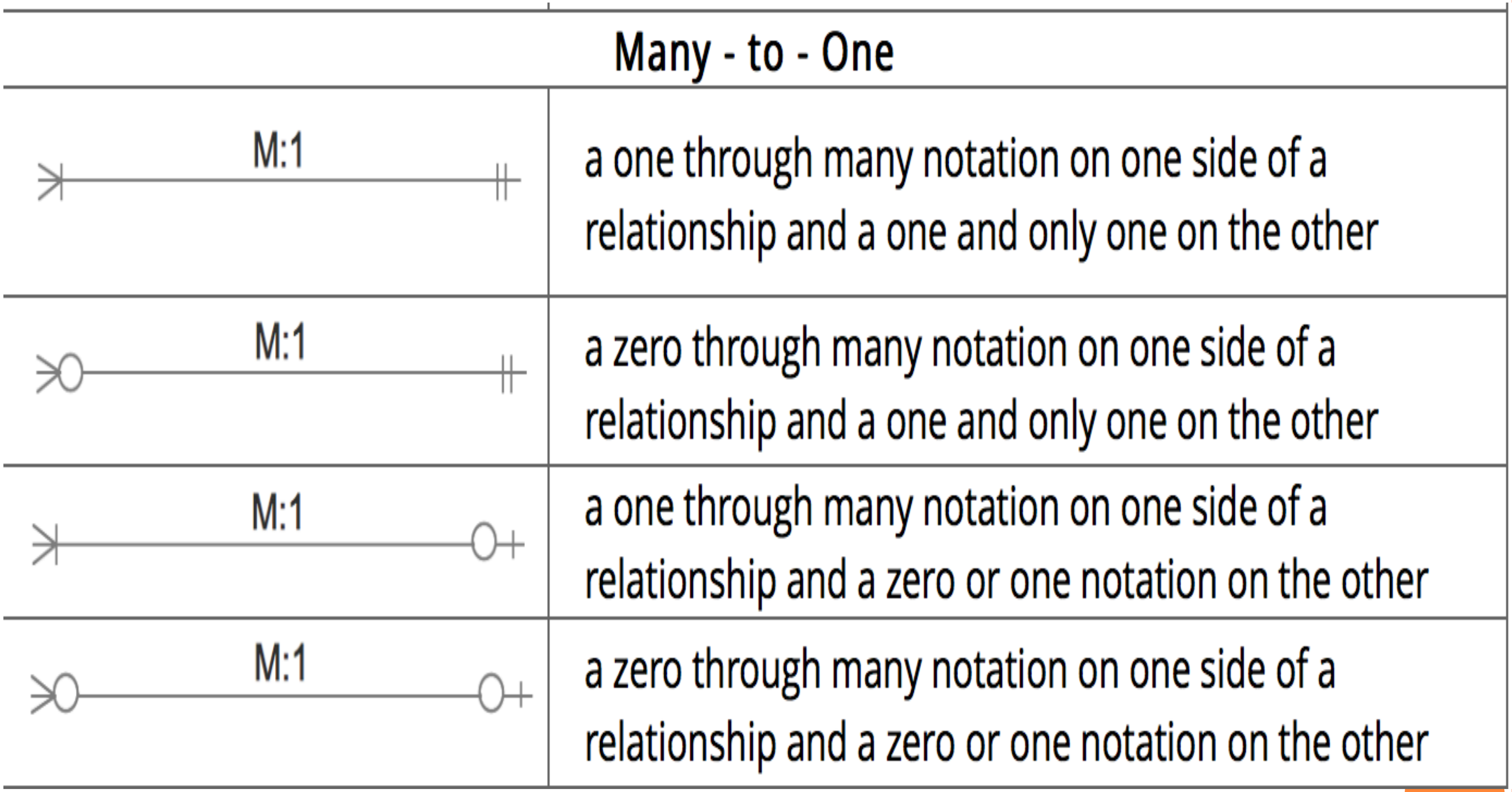
- Attribute



Crow’s Foot Notation -Cardinalities

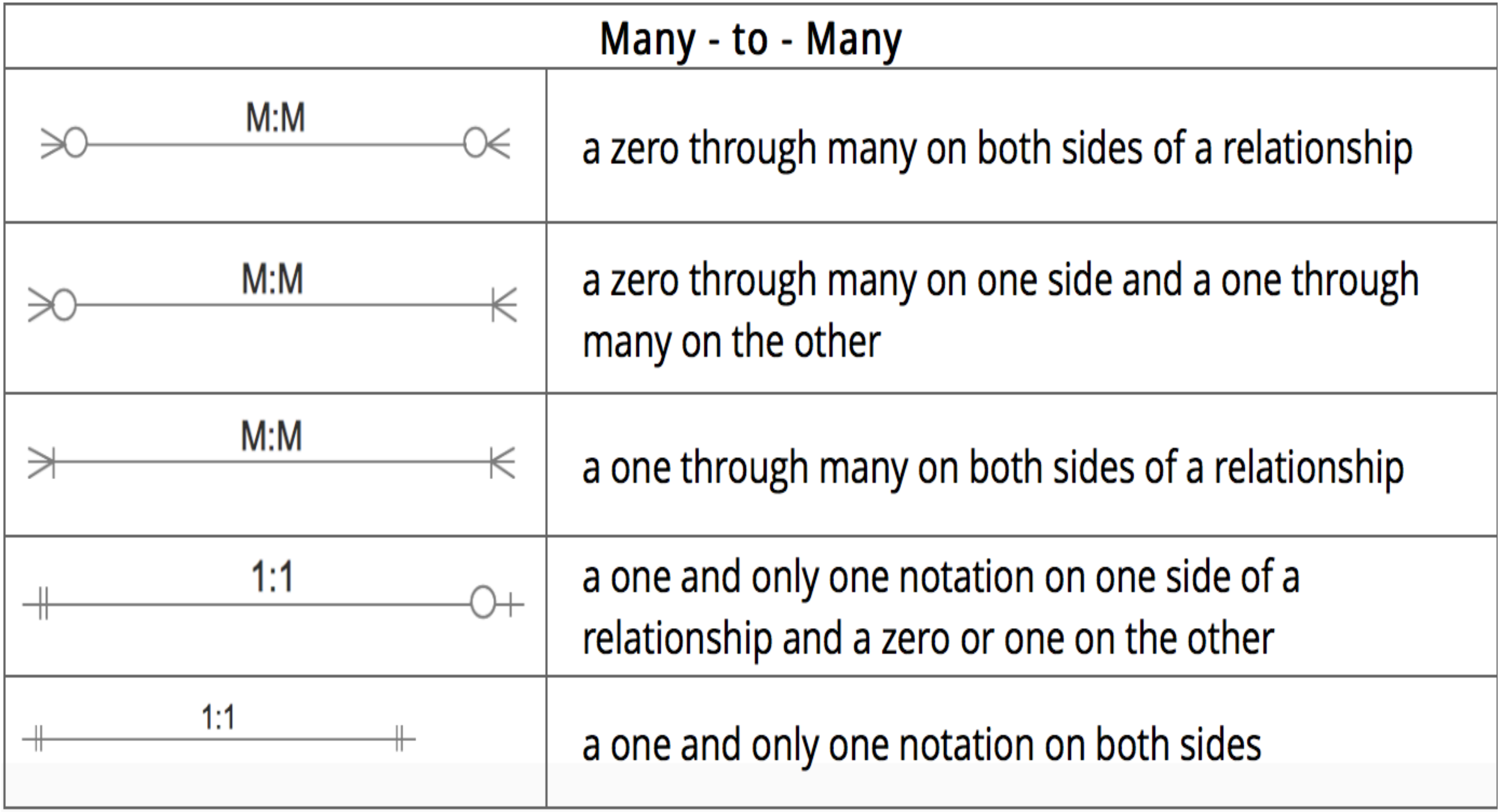
SYMBOL	MEANING	UML REPRESENTATION
	One and only one	1
	One or many	1..*
	Zero, or one, or many	0..*
	Zero, or one	0..1

Crow's Foot Notation -Cardinalities



Many - to - One	
M:1	a one through many notation on one side of a relationship and a one and only one on the other
M:1	a zero through many notation on one side of a relationship and a one and only one on the other
M:1	a one through many notation on one side of a relationship and a zero or one notation on the other
M:1	a zero through many notation on one side of a relationship and a zero or one notation on the other

Crow's Foot Notation -Cardinalities



Many - to - Many	
M:M	a zero through many on both sides of a relationship
M:M	a zero through many on one side and a one through many on the other
M:M	a one through many on both sides of a relationship
1:1	a one and only one notation on one side of a relationship and a zero or one on the other
1:1	a one and only one notation on both sides

ERD to Relational Model

Next we are going to look at how to map entities, attributes and relationships to the Relational Model

RELATIONAL MODEL

- A model of tables and fields
  - (lower level than E-R)
- Fields cannot be multivalued
- A Primary key is the unique identifier of the table, and can be 1 or more fields.
- A Foreign key is a field that is a primary key in another table.

Relational Model

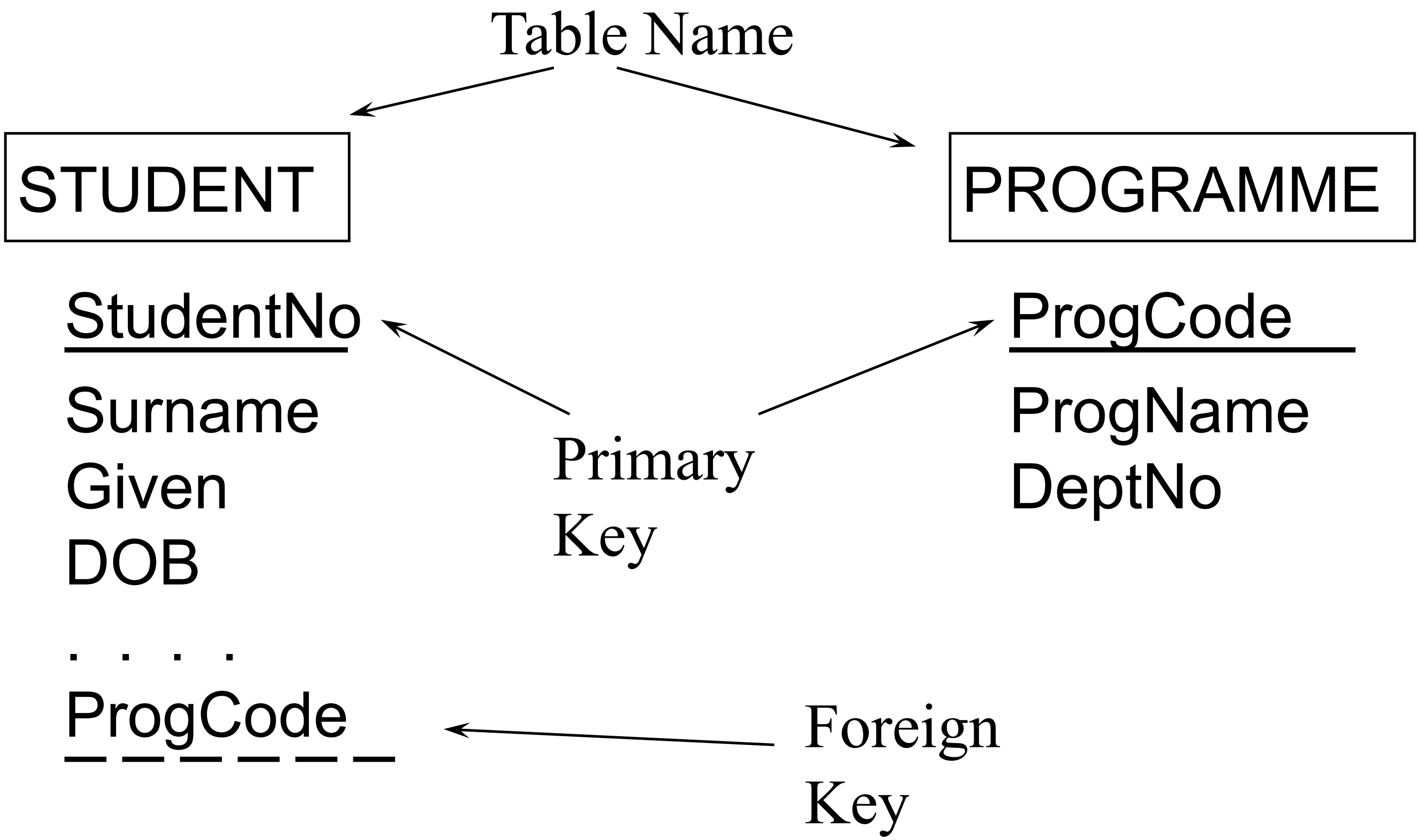


Table Name

STUDENT

StudentNo

Surname

Given

DOB

. . . .

ProgCode

Primary Key

PROGRAMME

ProgCode

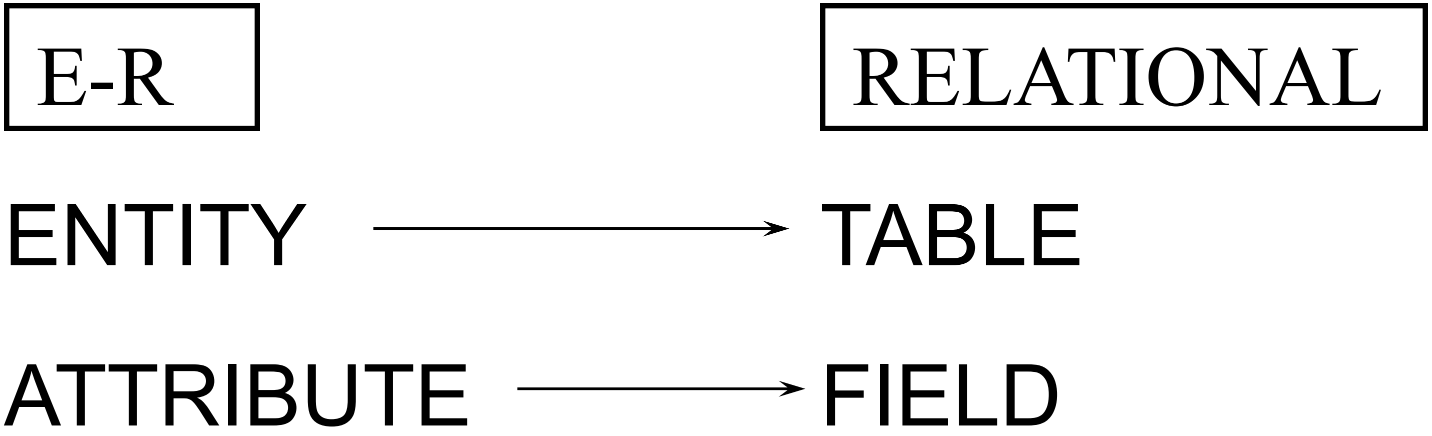
ProgName

DeptNo

Foreign Key

Mapping E-R and Relational

- Both models model the same thing, a database structure (existing or future)
- One model can be re-drawn to the other



E-R

RELATIONAL

ENTITY → TABLE

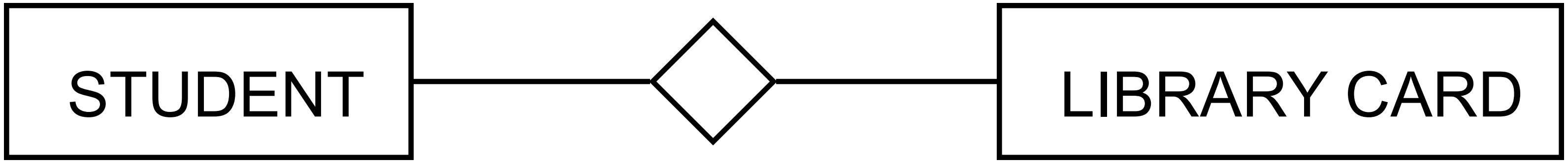
ATTRIBUTE → FIELD

Mapping - Relationships

- Relationships in the E-R model are represented by Foreign Keys in the relational model
- Foreign keys are NOT attributes in the E-R model, they represent the relationship diamond.
- One more time, foreign keys are NOT E-R attributes!

Mapping - 1:1 Relationship

- Put the foreign key on either side



STUDENT

StudentNo

Surname

LIB CARD

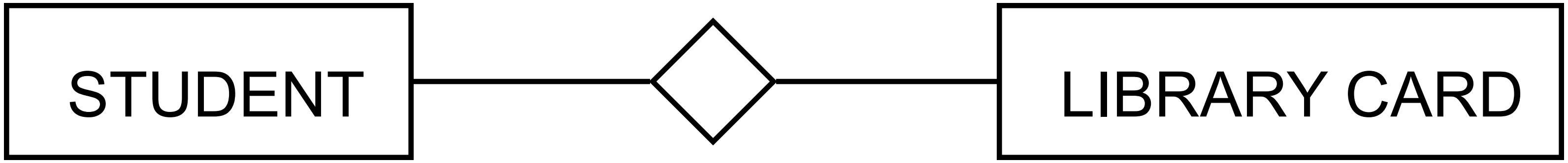
Barcode

Limit

StudentNo

Mapping - 1:1 Relationship

- OR Put the foreign key on other side
  NOT BOTH sides



STUDENT

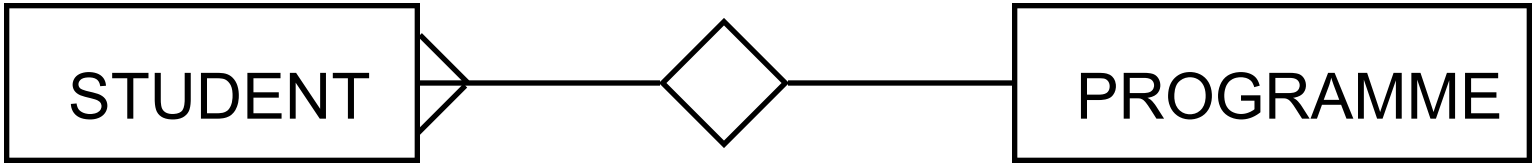
StudentNo
Surname
Barcode

LIB CARD

Barcode
Limit

Mapping - 1:M Relationship

- Put the foreign key on the Many side



STUDENT

StudentNo
Surname
. . . .
ProgCode

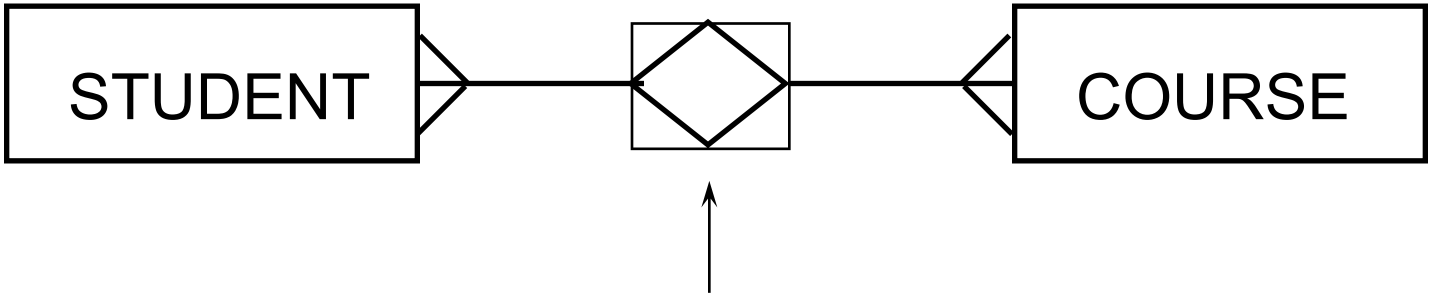
PROGRAMME

ProgCode
ProgName
DeptNo

Mapping - M:N Relationship

- Create a new table to represent the relationship
- Primary keys of both sides are combined to make the primary key of the new table.

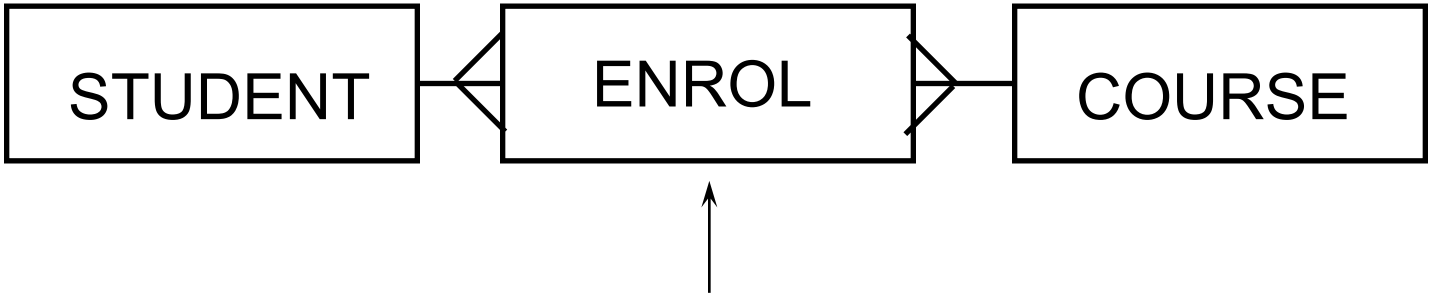
Mapping - M:N Relationship



STUDENT	ENROL	COURSE
StudentNo	StudentNo+	CrsCode
Surname	CrsCode	CrsName
	Result	DeptNo

NOTE: Gerund is halfway between a relationship and an entity.

Mapping - M:N Relationship (Alternative)



STUDENT

StudentNo

Surname

ENROL

StudentNo+

CrsCode

Result

COURSE

CrsCode

CrsName

DeptNo

Components - summary

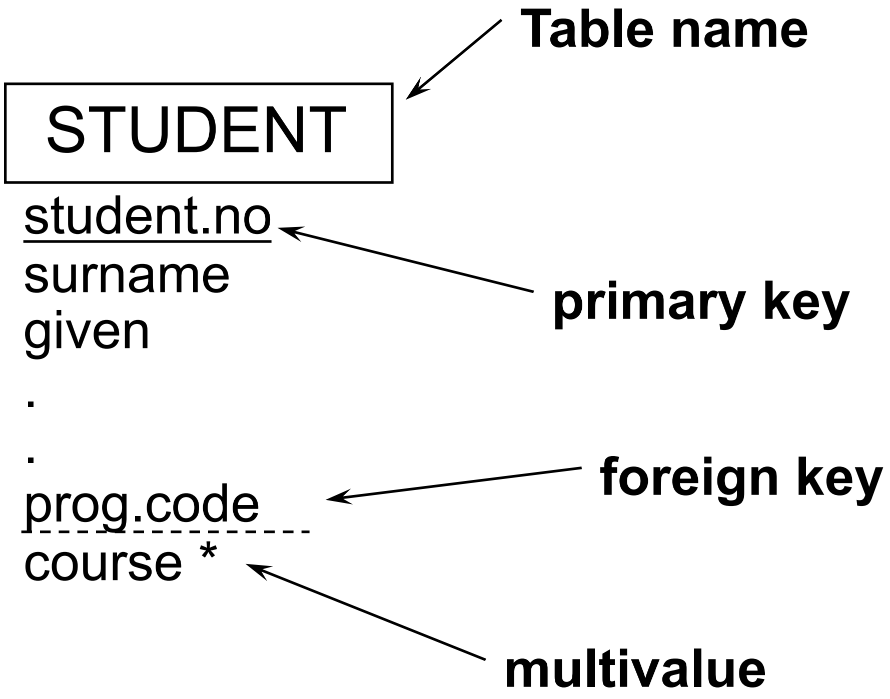


Table name

STUDENT

student.no

surname

given

.

.

prog.code

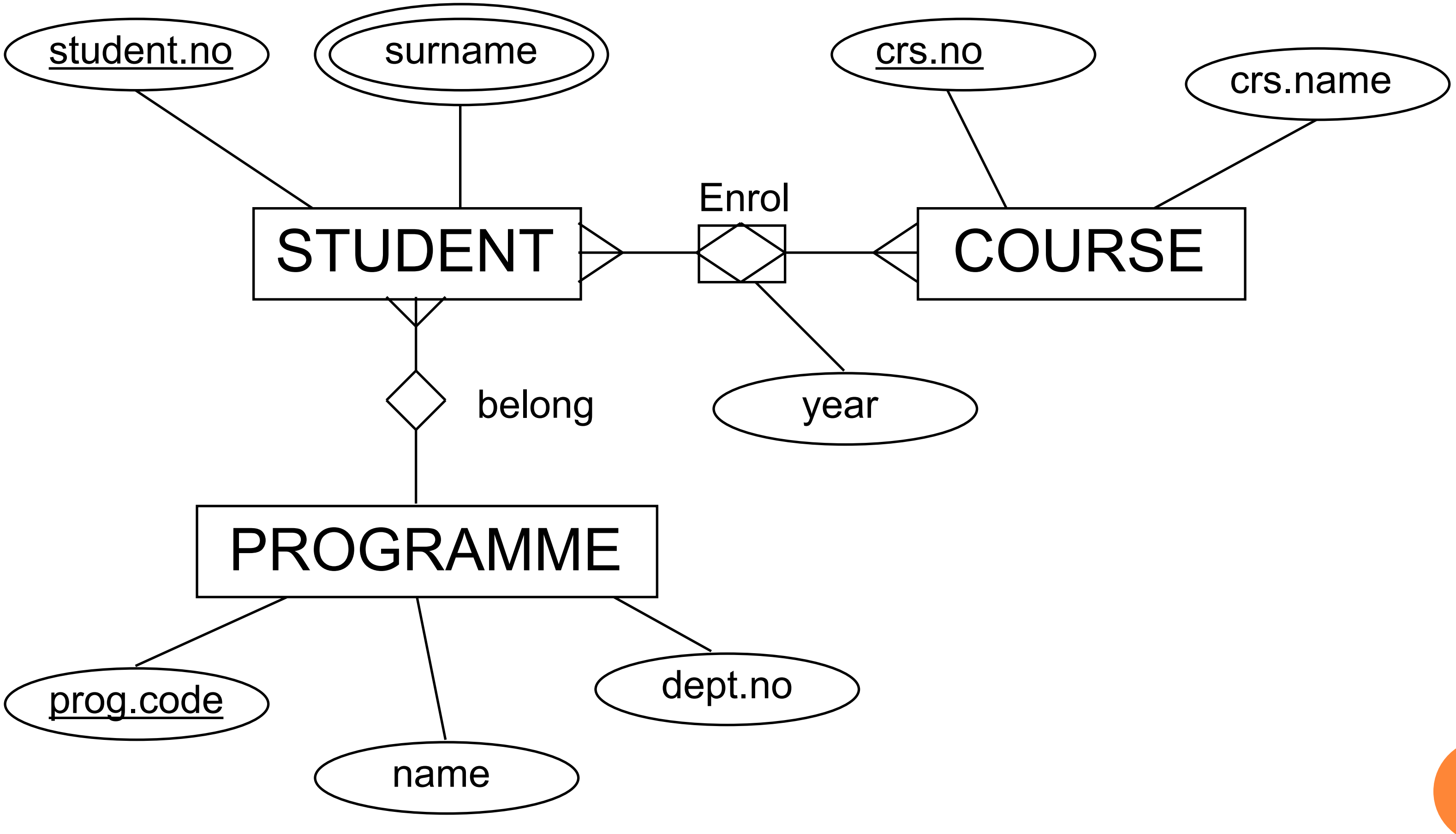
course *

primary key

foreign key

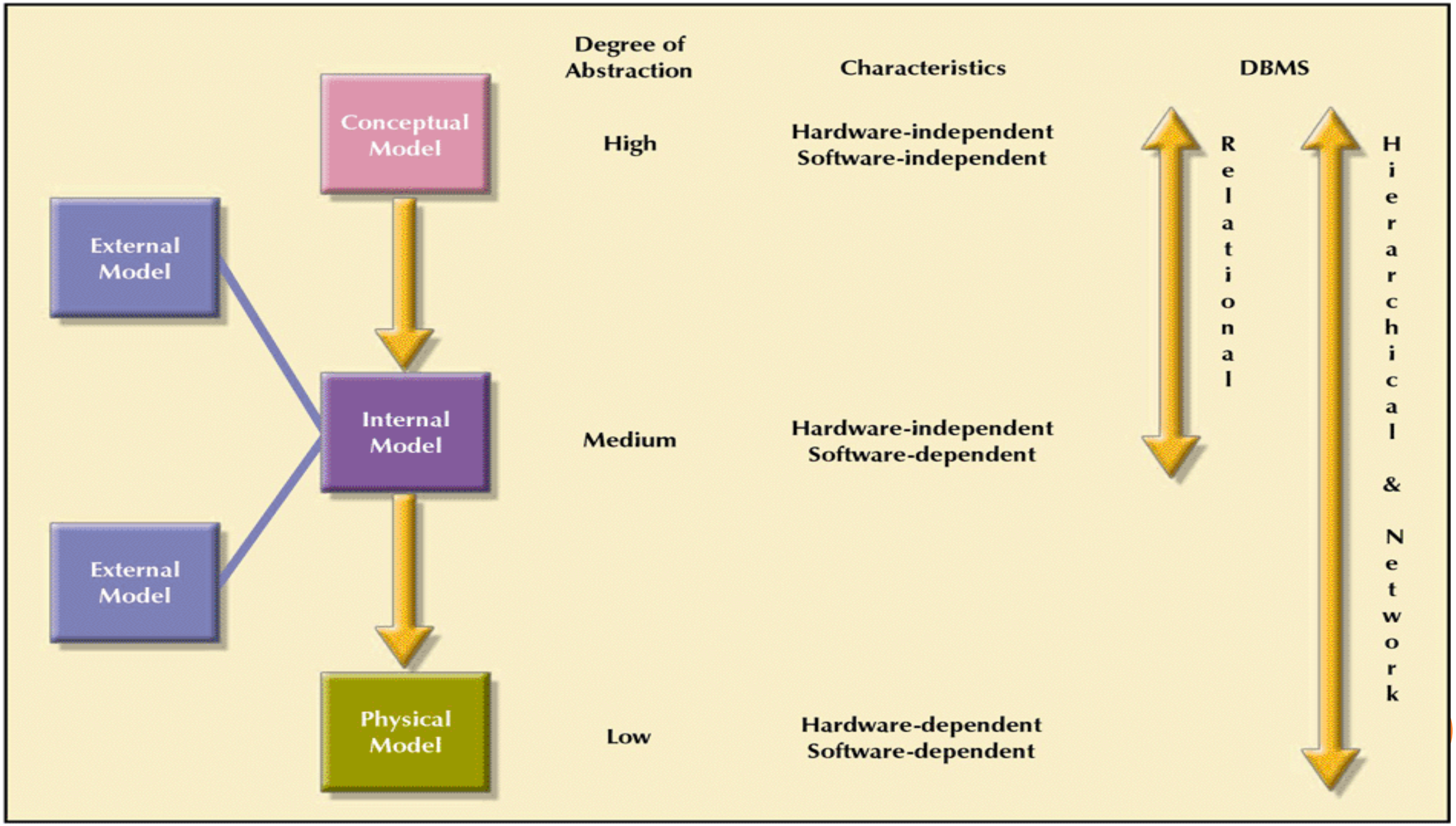
multivalue

Components - summary



Levels of data abstraction

FIGURE 2.10 Data Abstraction Levels

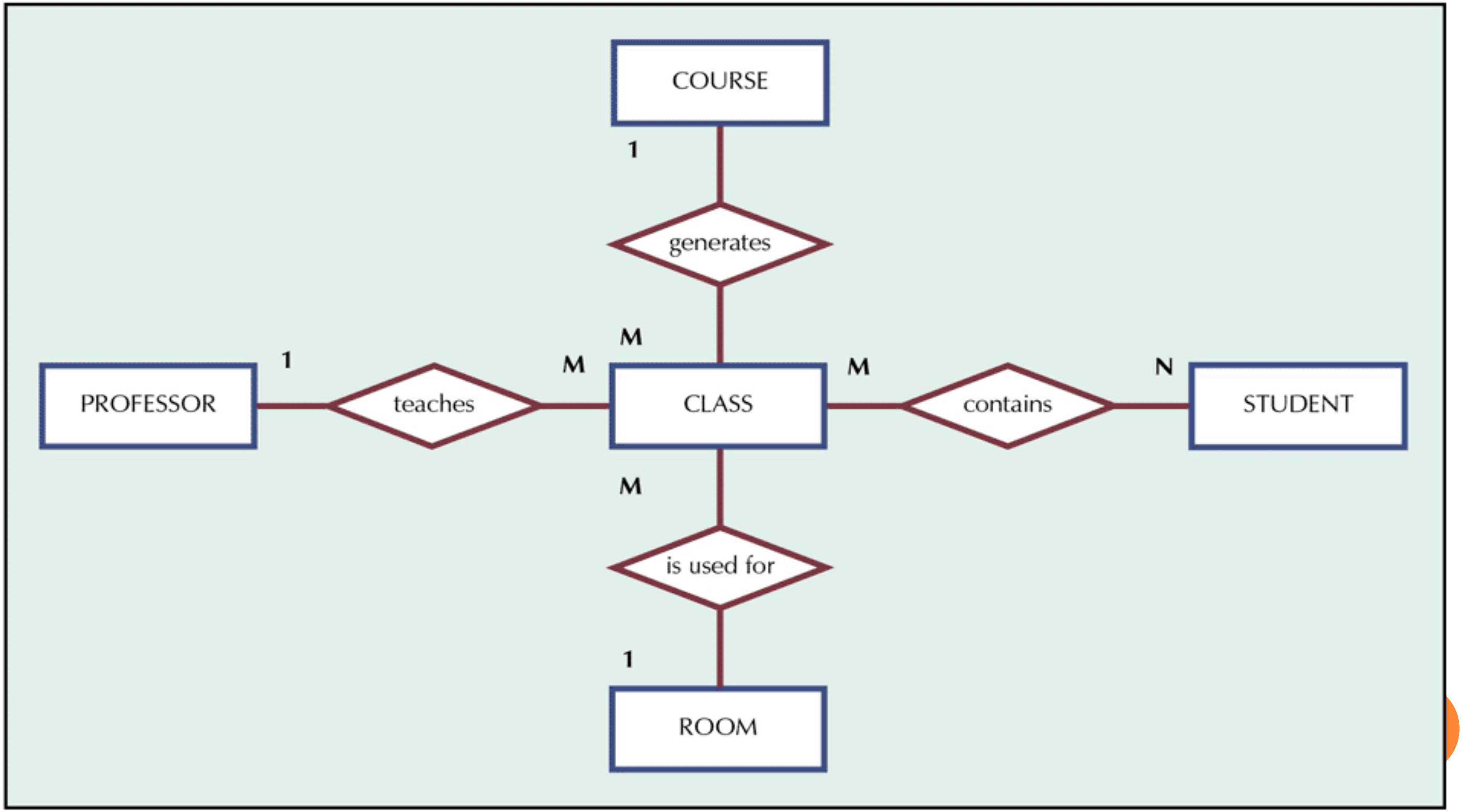


Conceptual model

- Gives a global view of the DB
- Enterprise wide, high level representation for managers
  - Models avoids details
  - Only main objects represented
- ER is the most commonly used model

Conceptual model: an example

FIGURE 2.12 A Conceptual Model for Tiny College



Advantages of conceptual models

- Easy to understand
  - Provides a macro view of the data environment
  - Independent of the software/hardware
- Why is the above important?

Internal model

- Represents what the DBMS “sees”
- Translates the conceptual model to the DBMS’s implementation
  - Software dependent
  - Hardware independent

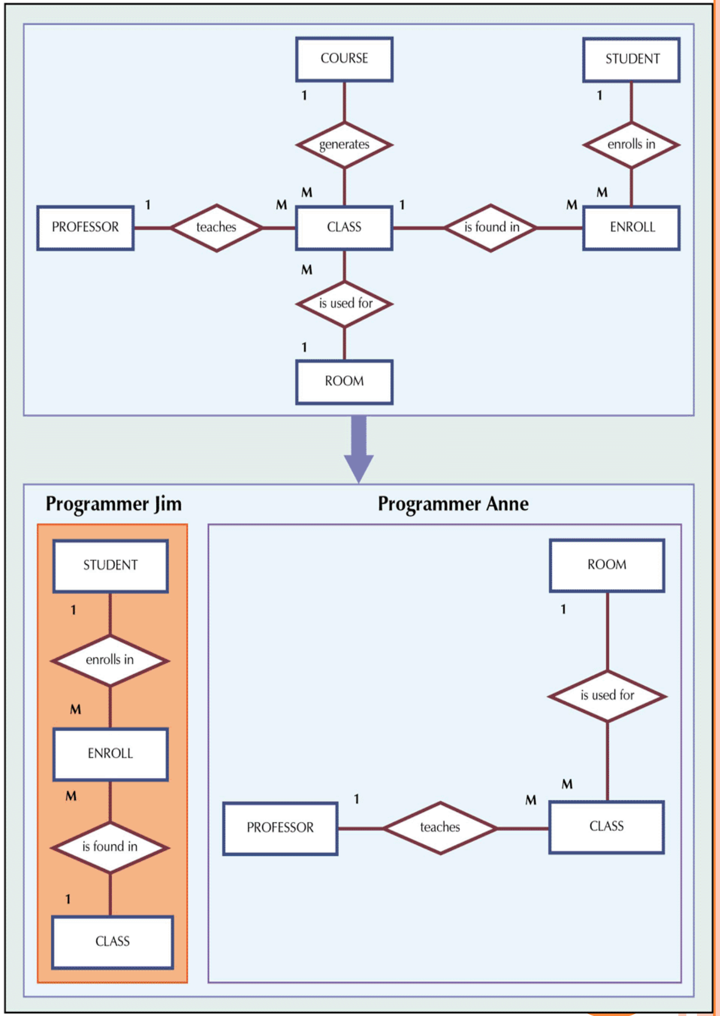
External model

- Provides the end-users' view of the data environment
- Provides a means to sub-divide requirements and constraints into functional modules
  - Very important for large databases
  - Like UML provides rules to help develop good databases

External model: an example

- Makes DB application development easier
  - Breaks down data requirements into manageable chunks
  - Provides a way to enforce security constraints
- DBMS dependent but hardware independent

FIGURE 2.13 A Division of an Internal Model into External Models



Physical model

- Operates at the lowest level of abstraction
  - Specifies how the data are stored
  - Software and hardware dependent
- The “data specialist” kind of work
  - Detailed knowledge of the hardware and software
  - Pretty much “standardize”

Data abstraction: a summary

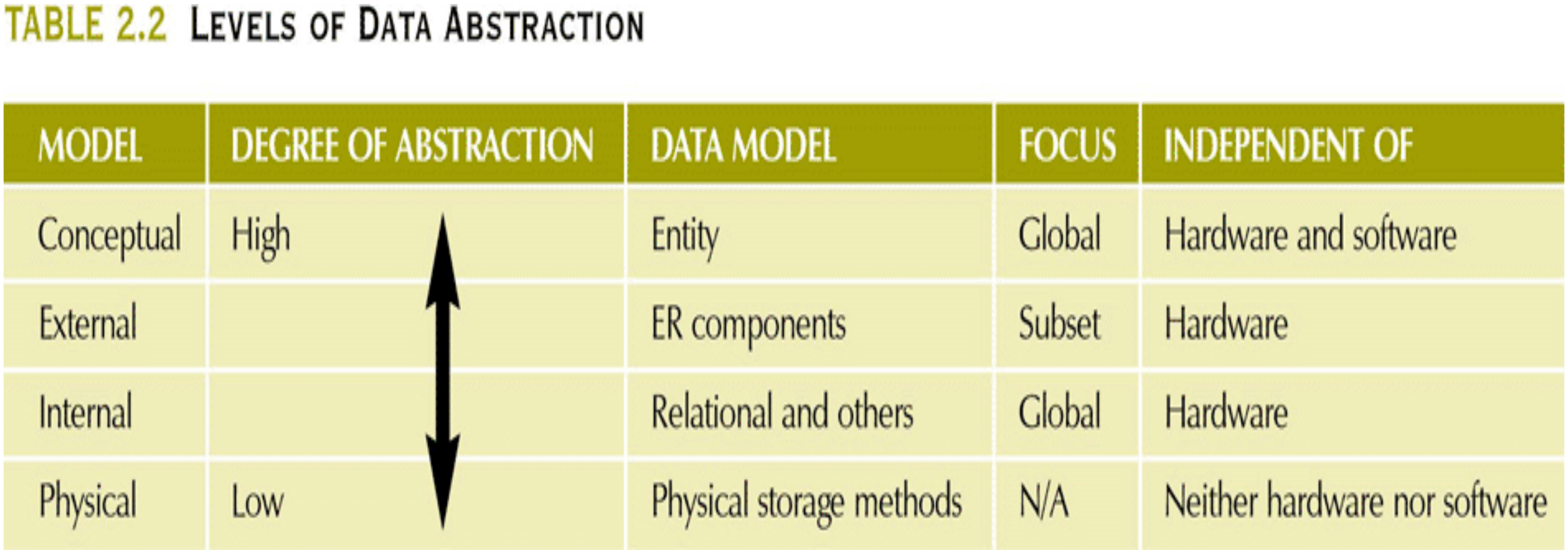


TABLE 2.2 Levels of Data Abstraction

MODEL	DEGREE OF ABSTRACTION	DATA MODEL	FOCUS	INDEPENDENT OF
Conceptual	High	Entity	Global	Hardware and software
External		ER components	Subset	Hardware
Internal		Relational and others	Global	Hardware
Physical	Low	Physical storage methods	N/A	Neither hardware nor software