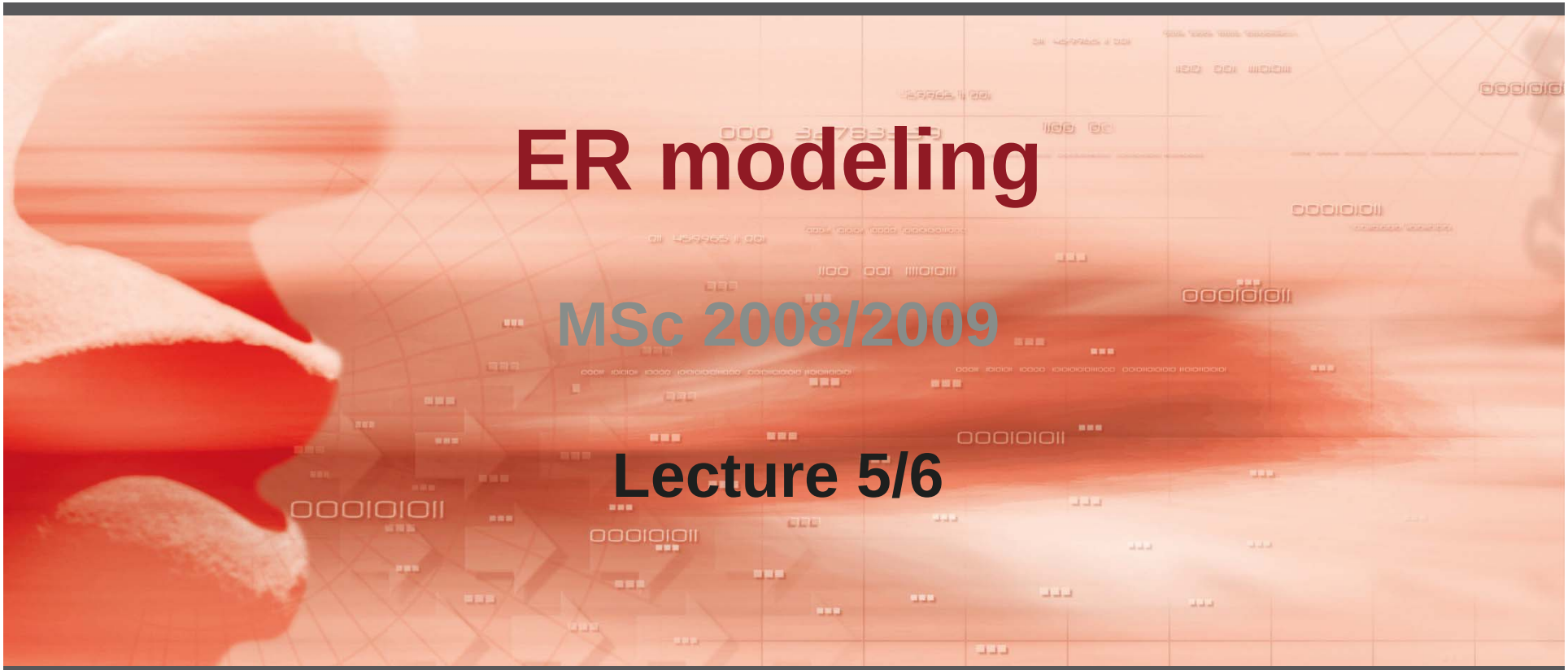


ER modeling

MSc 2008/2009

Lecture 5/6

Summary of the previous lecture

STI · INNSBRUCK

- General principles.
- Basic elements and their properties: entities, attributes, associations, specialization/generalization.
- Relationships and their characteristics.

Today‘s lecture

STI · INNSBRUCK

- **ER modeling**

Slides based on „Introduction to Entity-relationship modeling“ at
http://www.inf.unibz.it/~franconi/teaching/2000/ct481/er-modelling/

Databases

STI · INNSBRUCK

- Data bases and data base management system – the core information systems technology
- Example of uses: storing corporate data, web pages, on-line movies, work flow information, documents
- Databases are one the most applied solutions to business process reengineering (BPR)

ER Modeling

STI · INNSBRUCK

- Purpose of ER modelling: to create an accurate reflection of the real world in a database
- The ER model doesn’t actually give us a database description; it gives an intermediate step from which it is easy to define a database

Example

STI · INNSBRUCK

- Every department within our company is in only one division. Each division has more than one department in it. We don’t have an upper limit on the number of departments that a division can have. For example, the *New Business Development*---the one managed by Mackenzie---and *Higher Education* departments are both in the *Marketing* division.

Definitions

STI · INNSBRUCK

- An *entity type* is a collection of entities that share a common definition.
- An *entity* is a person, place, concept, or thing about which the business needs data.
- Example:
  - *Department* is the name of one *entity type*.
  - One instance of this entity type is the *New Business Development* department.

Definitions

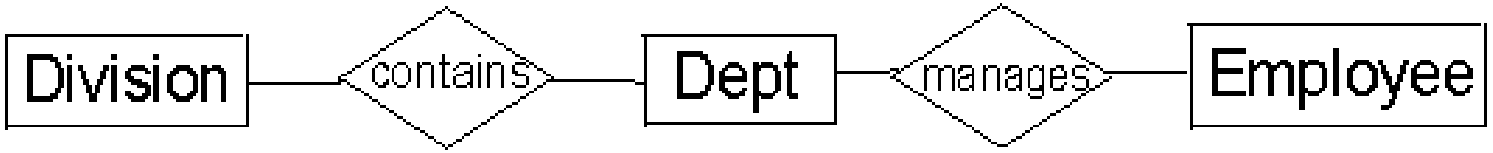
STI · INNSBRUCK

- A *relationship* is an association between entity types.
- The defining characteristic of a relationship is that several entity types are involved.
- Example: Three entity types (*Employee*, *Department*, *Division*) and two relationships among these entity types (*manages*, *contains*).

Graphical representation

STI · INNSBRUCK

- ER models are usually represented graphically.
- The language we are going to use represents entity types as *rectangles* and relationships as *diamonds*.
- Example:

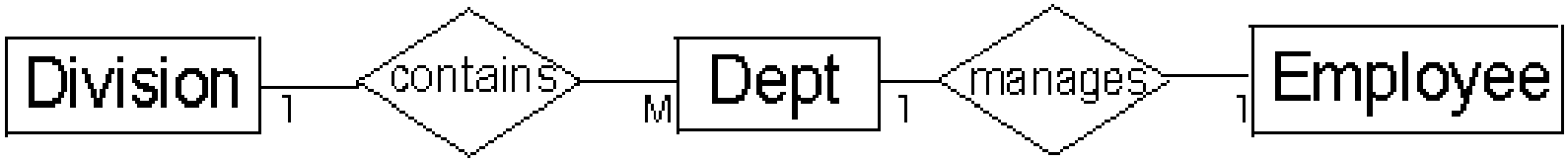


Relationships

- No direct association between division and employee. *This does not mean that there is **no** relationship between division and employee.*
- An ER diagram should contain the minimum number of relationships necessary to reflect the situation.

Relationships

- Example: divisions have multiple departments and departments can only be contained within one division.
- Or, for every *one* division there can be *many* departments. In the language of ER modelling this is called a **1:M** (read: “one to many”) relationship.
- Example:



Cardinality

STI · INNSBRUCK

- A relationship's *cardinality* defines the maximum number of entities of one type that can be associated with an entity of another type.
- 1:1 One entity of type *X* can be associated with, at most, one entity of type *Y*. One entity of type *Y* can be associated with, at most, one entity of type *X*.

Cardinality

STI · INNSBRUCK

- 1:M One entity of type *X* can be associated with, at most, one entity of type *Y*. One entity of type *Y* can be associated with, at most, one entity of type *X*.
- M:M One entity of type *X* can be associated with many entities of type *Y*. One entity of type *Y* can be associated with many entities of type *X*.

Existence

STI · INNSBRUCK

- A relationship's *existence* defines what we know about the existence of any entity on the other side of a relationship from a given entity.
- Existence is given as
  - *optional*,
  - *mandatory*,
  - or *unknown*.

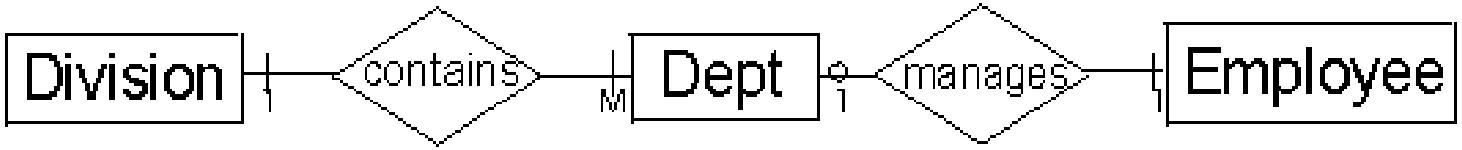
Existence

STI · INNSBRUCK

- Example
  - optional
    - A department need not have any manager.
  - mandatory
    - A department must have at least one manager.
  - unknown
    - It is unknown whether or not a department has to have a manager.

Existence

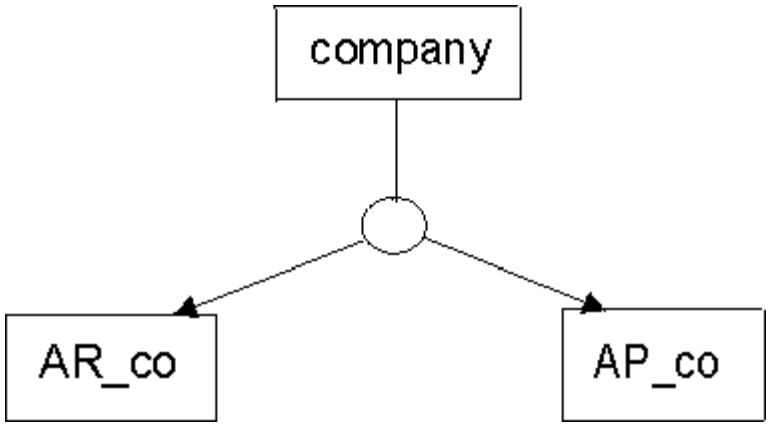
- Given any (randomly chosen) department, there must be an employee on the other side of the *manage* relationship. Thus, the relationship is *mandatory* in this direction. This is indicated by a dash on the line.
- Given any (randomly chosen) employee, there need not be any department on the other side of the *manage* relationship. Thus, the relationship is *optional* in this direction. This is indicated by a circle on the line.



Entity subtypes

STI · INNSBRUCK

- An *entity subtype* is a collection of entities of the same type to which a narrower definition and additional attributes and/or relationships apply.
- Example:



Entity subtypes

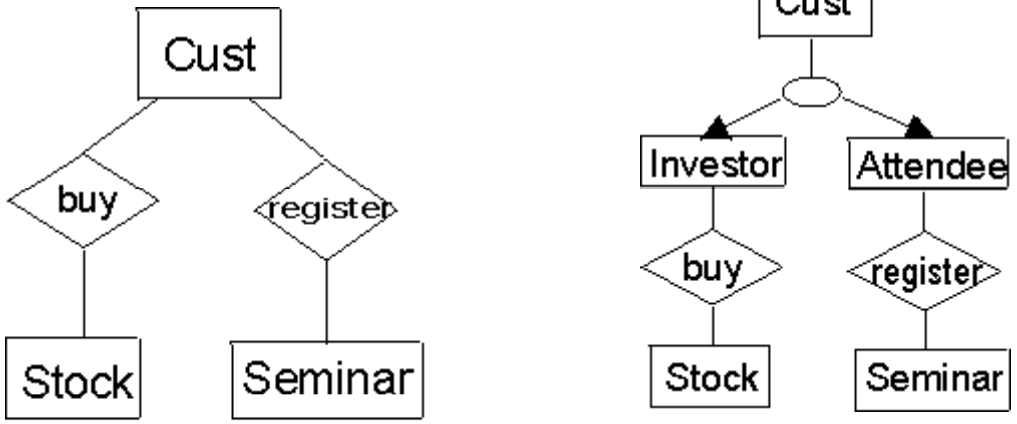
STI · INNSBRUCK

- There are many situations in which subtypes can be created but should not be. Only create subtypes
  - if the subtype is involved in relationships that the other subtypes are not or
  - if the subtype needs to have additional facts stored with it.

Entity subtypes

STI · INNSBRUCK

- Example



- If a relationship *defines* the members of a proposed subtype, then use the relationship instead of the subtype.

Attributes

STI · INNSBRUCK

- An *attribute* is a descriptor whose values are associated with individual entities of a specific type.
- The attribute value for any single entity can have only one value at a given time. This value can change over time.
- Example:
  - An attribute of an employee might be *salary*. At any one time if you asked for the salary level of a certain employee, then you should get one answer.

Identifier

STI · INNSBRUCK

- A identifier uniquely identifies a single (at least one, and no more than one) entity.
- If you know the value of the identifier, then you know exactly which entity you are dealing with.
- The identifier's value will never change over time.

Degree of a relationship

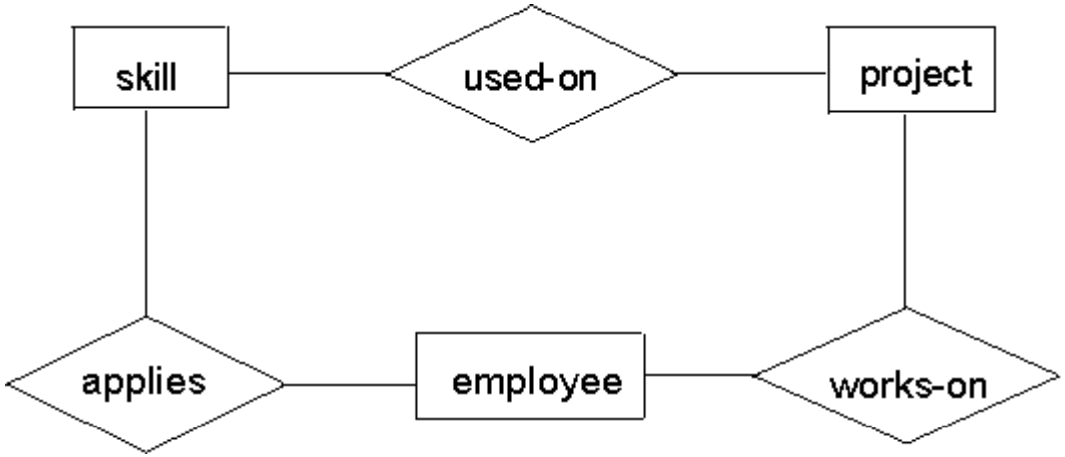
STI · INNSBRUCK

- Relationships can be classified by the number of entity types involved. This is referred to as the *degree* of a relationship.
- Common degrees of relationships:
  - binary
    - This is a relationship between two entity types.
  - ternary
    - This is a relationship between three entity types.
  - recursive
    - This is a relationship involving only one entity type.

Ternary relationships

STI · INNSBRUCK

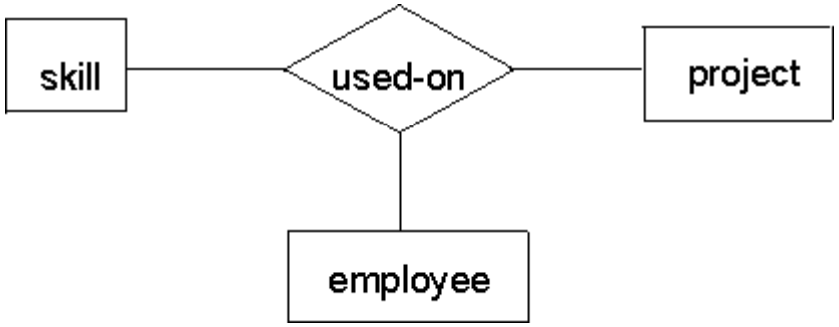
- Example:
  - Suppose you know the following:
    - works-on
      - Lindsey and Mackenzie have worked on projects *A* and *B*.
    - applies
      - Lindsey has used skills *interface design* and *database design* while Mackenzie only used her *database design* skill.
    - used on
      - Both skills have been used on both projects.
  - Figure out on which projects Lindsey used which skills.



Ternary relationships

STI · INNSBRUCK

- In order to capture the necessary information the database needs a ternary relationship.

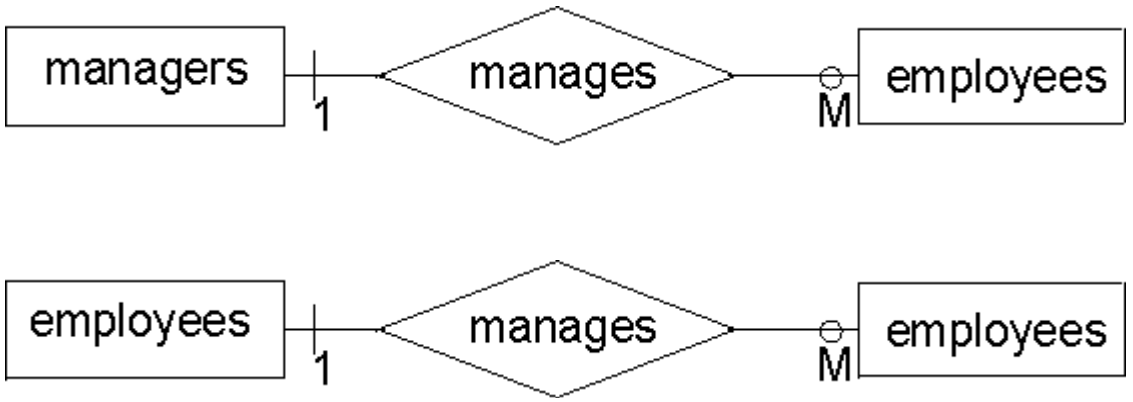


- The *used on* relationship captures information three pieces at a time. It stores facts such as:
  - Lindsey used interface design skill on project *A*.
  - Mackenzie used database design skill on project *A*.

Recursive relationship

STI · INNSBRUCK

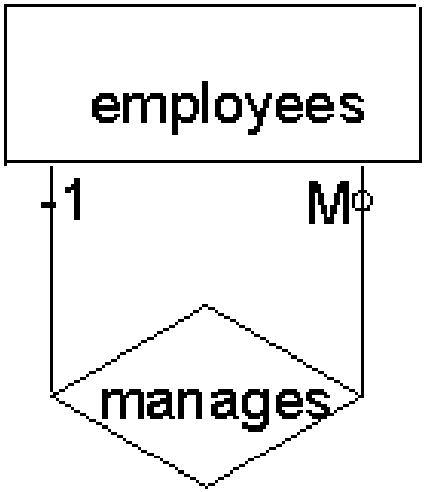
- A recursive relationship is a relationship that an entity has with itself.
- Example:
  - An employee who is the manager of other employees.
  - A manager is really just another employee



Recursive relationship

STI · INNSBRUCK

- Reading from left, down, and back up: An employee may not manage any other employees but may manage many.
- Reading from right, down, and back up: All employees are managed by exactly one other employee.



Attributes of a relationship

STI · INNSBRUCK

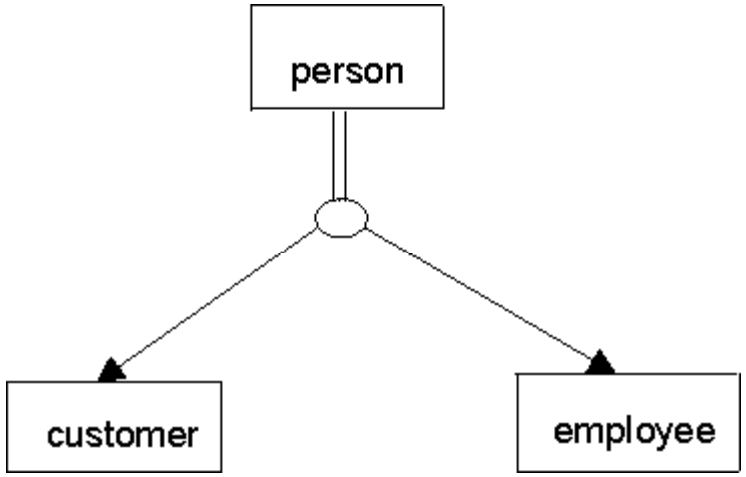
- Relationships can also have attributes
- Example:

  - Store when a person has joined a club
    - If the attribute is of the *person* entity, then this would indicate when the person joined a club but we would not know which club.
    - If the attribute is of the *club* entity, then this would indicate (possibly) when the club was founded or (possibly) when the most recent member joined the club but we would not know the dates on which each person joined.
  - Solution: make *join date* an attribute of the *is member* relationship.

Entity subtype partitioning

STI · INNSBRUCK

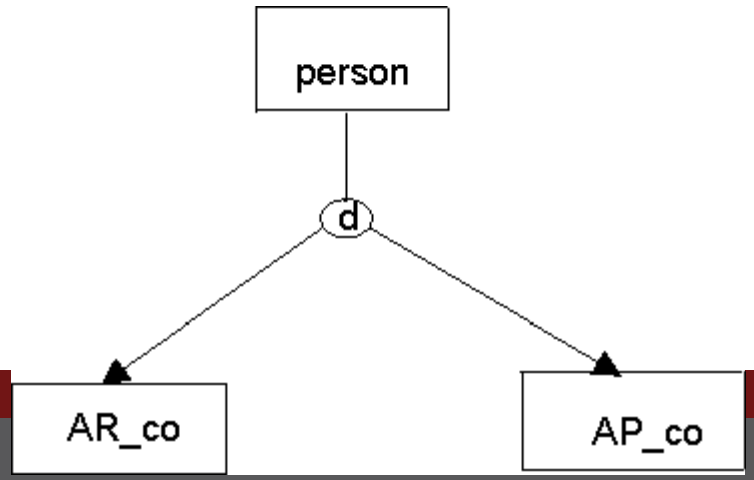
- **Optional versus mandatory**
  - mandatory
    - He/she can demand that the person be classified as one of the subtypes.
  - optional
    - He/she can allow a person to be created without classifying the person as any subtype.
  - Neither one is preferable to the other. The proper one to choose depends on the business situation.



Entity subtype partitioning

STI · INNSBRUCK

- **Disjoint versus overlapping**
  - disjoint
    - If entities are allowed to be no more than one subtype, then the subtypes are said to be *disjoint*.
  - overlapping
    - If entities can be classified as several subtypes, then the subtypes are said to be *overlapping*.



Aggregation of entity types

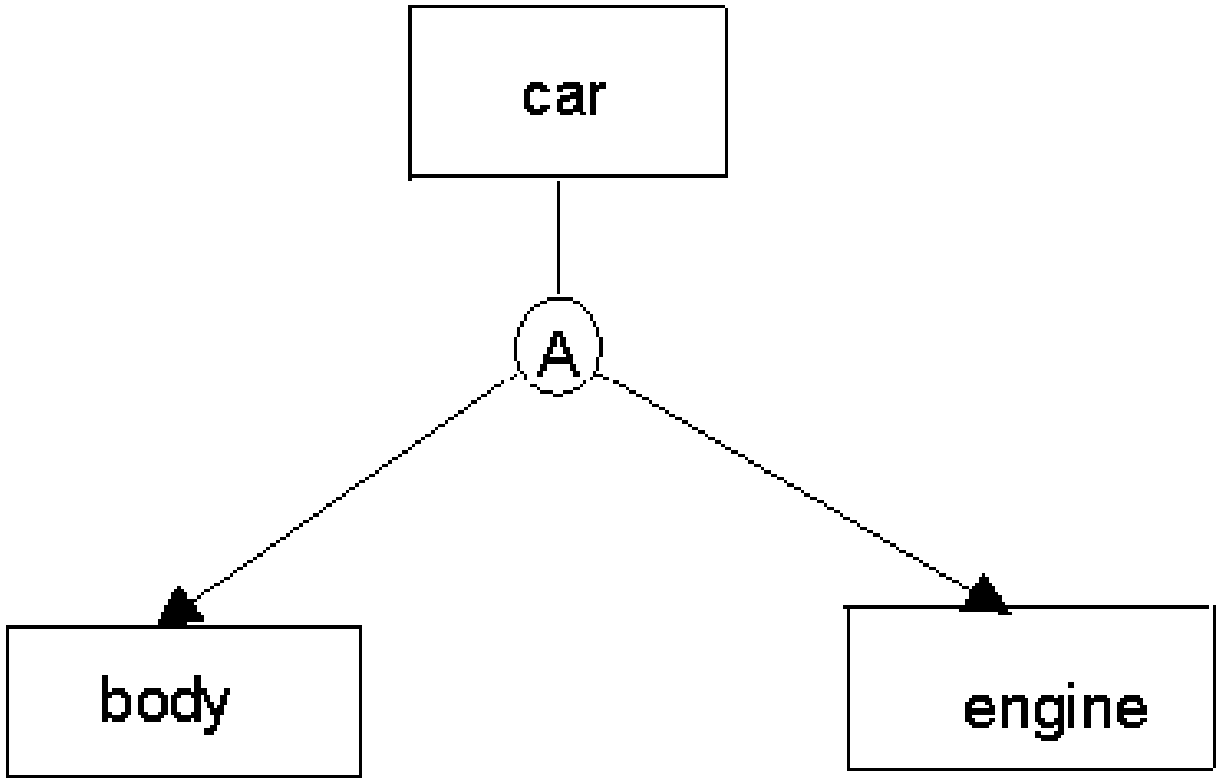
STI · INNSBRUCK

- Subtypes are generally thought of in terms of *X is a Y* (which is why these are commonly referred to as *is-a* relationships).
- Another type of relationship that needs to be represented in a database is the *part of* relationship, more formally called *aggregation*.

Aggregation of entity types

STI · INNSBRUCK

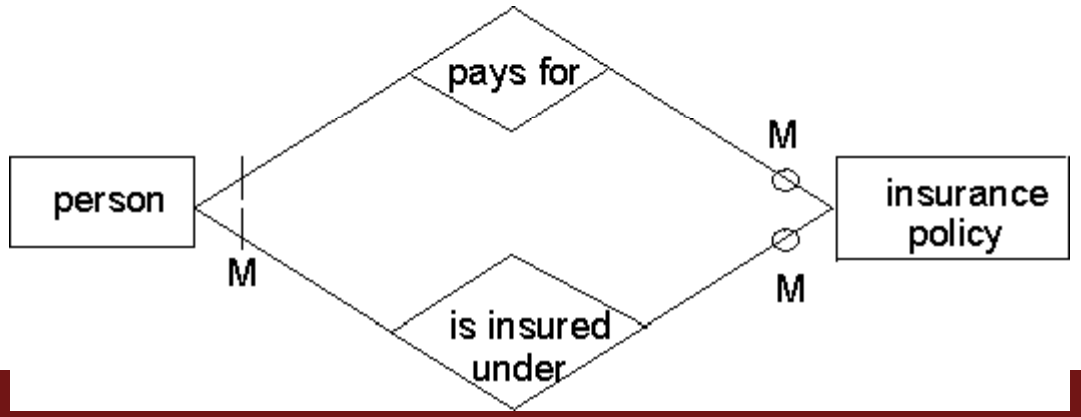
- Example



Parallel relationships

STI · INNSBRUCK

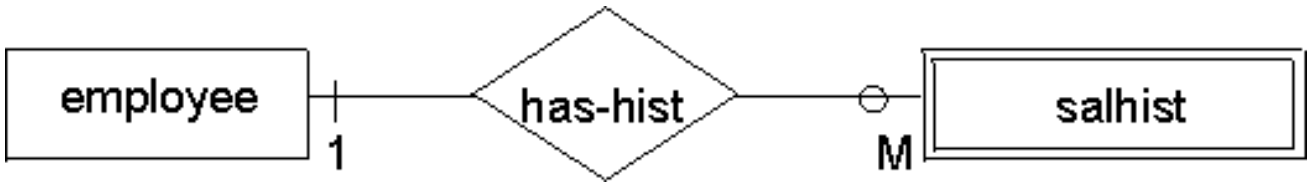
- Two entities can have more than one type of relationship
- Example
  - the entity types *person* and *insurance policy* and the relationships between them of *pays for* and *is insured under*.



Weak entities

STI · INNSBRUCK

- Weak entities are entities, but with a difference – weak entities only exist because some other entity exists.
- Example:



Types of attributes

STI · INNSBRUCK

- Basic
  - These are values provided to the business. These are the types of attributes that we have been discussing so far. Think of *name*, *address*, etc. These values cannot be deduced from the values of other attributes.
- Designed
  - This is invented and exists only in the database. An example might be a unique identifier for a department. This value is not changed once it is set.
- Derived
  - This is a value that can be calculated from the value of other attributes in the database. An example might be the age of an employee when the birth date is in the database. These attributes should, generally, not be stored in the database but should be calculated when needed.

Attribute optionality

STI · INNSBRUCK

- Not all entities have a value for every attribute; however, some attributes must have a value for all entities.
- optional
  - An entity need not have a value associated with an optional attribute.
- mandatory
  - An entity must have a value associated with a mandatory attribute.

Attribute optionality

STI · INNSBRUCK

- Example
  - an *employee* entity type has attributes *hire date* and *termination date*. *Hire date* would certainly be classified as a mandatory attribute; if the employee didn't have a hire date, then the person couldn't very well be an employee.

- The optionality of an attribute depends highly on the business situation.

More about attributes

STI · INNSBRUCK

- default
  - This is the value that an attribute should take if it is not assigned a value.
- permitted range
  - These are the values that an attribute is allowed to take.
- composite
  - A composite attribute is an attribute made up of many other attributes.

More about attributes

STI · INNSBRUCK

- Examples
  - default
    - the state field of an employee table might have the default value of MI.
  - permitted range
    - the sale_price field of the inventory table might have a permitted range of sale_price > 0
  - composite
    - an employee's address that is made up of the house number, street, city, state, and zip

Assignments 5/6

STI · INNSBRUCK

a) What is the cardinality and existence of each of the following relationships in just the direction given? State any assumptions you have to make.

1. Husband to wife
2. Student to degree
3. Child to parent
4. Player to team
5. Student to course

b) For each of the following pairs of rules, identify two entity types and one relationship. State the cardinality and existence of the relationship in each case. If you don't think enough information is available to define either of these, then state an assumption that makes it clear. Draw the ER diagram.

1. A department employs many persons. A person is employed by, at most, one department.
2. A manager manages, at most, one department. A department is managed by, at most, one manager.
3. An author may write many books. A book may be written by many authors.
4. A team consists of many players. A player plays for only one team.
5. A lecturer teaches, at most, one course. A course is taught by exactly one lecturer.
6. A flight-leg connects two airports. An airport is used by many flight-legs.
7. A purchase order may be for many products. A product may appear on many purchase orders.
8. A customer may submit many orders. An order is for exactly one customer.

Asignments 5/6 (cont)

STI · INNSBRUCK

c) For each of the following sets of sentences, draw the corresponding ER diagram.

1. An account can be charged against many projects, though it may not be charged against any. A project must have at least one, though it may have many, accounts charged against it.
2. Projects *must* be classified as either top secret or civilian (but not both). There is information specific to top secret projects and specific to civilian projects that we want to record.
3. An employee must manage exactly one department. A department may or may not have one employee manage it.
4. Men are only allowed to supervise men. Women are only allowed to supervise women. We do not want to allow the database to hold data representing a man supervising a woman. An employee, regardless of sex, is assigned to exactly one office, with each office having exactly one employee in it. (Be sure to include the office entity in this diagram.)

Asingments 5/6 (cont)

STI · INNSBRUCK

d) Draw the corresponding ER diagram for the set of sentences below.

1. Each supplier has a unique name.
2. More than one supplier can be located in the same city.
3. Each part has a unique part number.
4. Each part has a color.
5. A supplier can supply more than one part.
6. A part can be supplied by more than one supplier.
7. A supplier can supply a fixed quantity of each part.

Thank You!

Questions?