Database Design in the 60's

- In the 60's

  database design = database implementation

- Specifications were block diagrams and record structures.
- No design methodology existed.

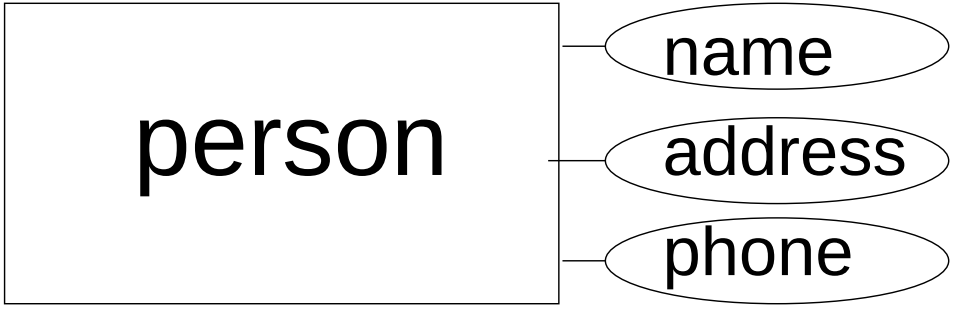
Lessons Learned

- There are plenty of databases around that are functionally inadequate;
  their documentation is sketchy and their maintenance is painful.
- *Lesson*: Design methodology is crucial to the long-term success of information systems.

Conceptual Schema

- 1976: Peter Chen proposed the *Entity-Relationship (ER) Model*.
- The ER Model is used to specify a conceptual schema; its main conceptual primitives are
  - *Entities* and
  - *Relationships*.
- Later it was extended: the *Extended ER Model* or *EER Model* to include generalization.

- **Entity**: real world *thing* or *object* (concrete or abstract).
- **Attributes**: important properties of entities.



- *Domain* or *type* (*Value Set*) of attribute:
  set of possible values of the attribute.
  As a compromise, the domain is usually extended with a special NULL value to cover the inapplicability of an attribute to some instances.

- *Instances* of an entity form an *entity set*.

  Each person in our world has a name, an address, and a phone number; he/she is an *instance* of the entity *person*.

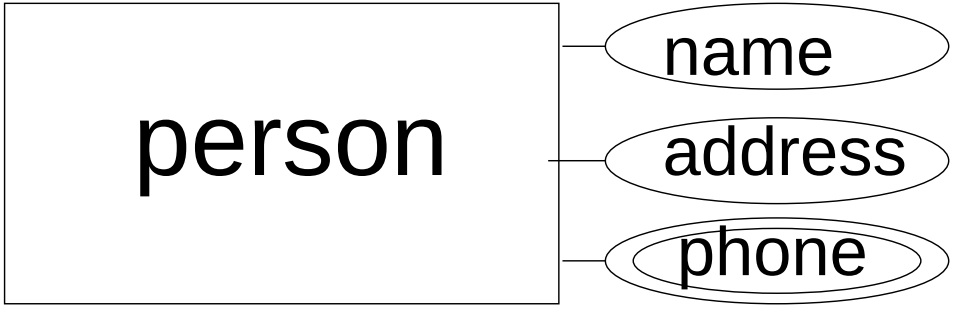
- Entity sets of distinct entities are disjoint.

Entity Instance

- An *instance* of an entity has values for each attribute.
- An *entity type* is implicitly defined by the *types* of the attributes.
- Each *instance* of an entity must be a member of its entity type.

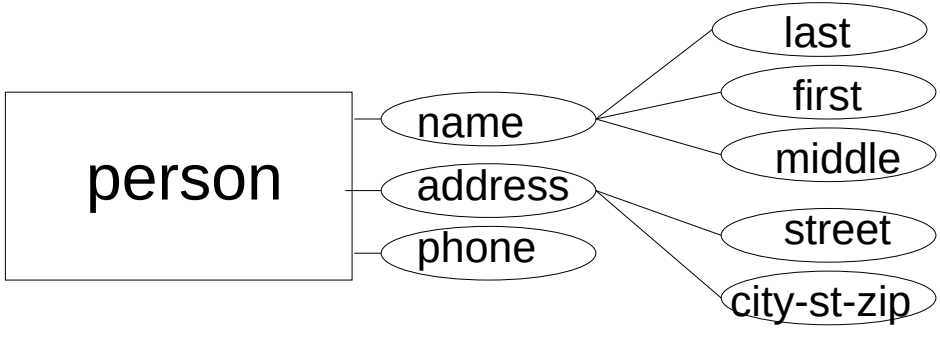
Single/Multi-valued Attributes

- *Single-valued* attribute: each entity instance has exactly one value for such an attribute.
- *Multi-valued* attribute: ... has ≥ 1 value ...



Composite Attributes

- Attributes can also be composite.
- *Composite* attribute can be divided into more basic attribute components with independent meanings.



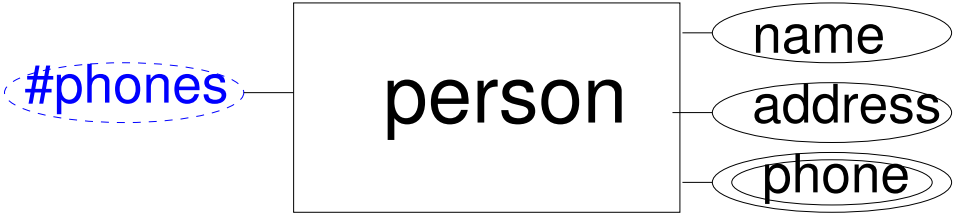
Composite Attributes

- are very different from multi-valued attributes.
- The multi-valued attributes we saw earlier were *(atomic)* or (*simple*).

  And so were single-valued attributes.

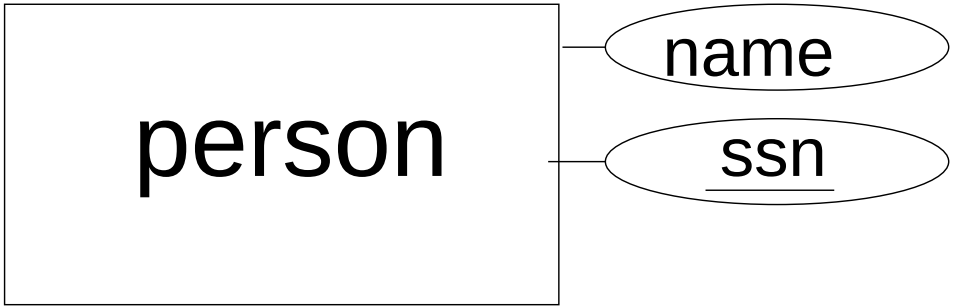
Derived Attributes

- *Derived* attribute: value can be computed from the ordinary (or *stored*) attributes.



Identifying (Key) Attributes

- *Key* attribute(s) serves to *identify* entities. Denote by underlining.
- No two instances of an entity can have the same value for its key attribute(s).



Identifying (Key) Attributes

- Can two attributes together form the key?

Identifying (Key) Attributes

- Can two attributes together form the key?

  Yes. Both are underlined.

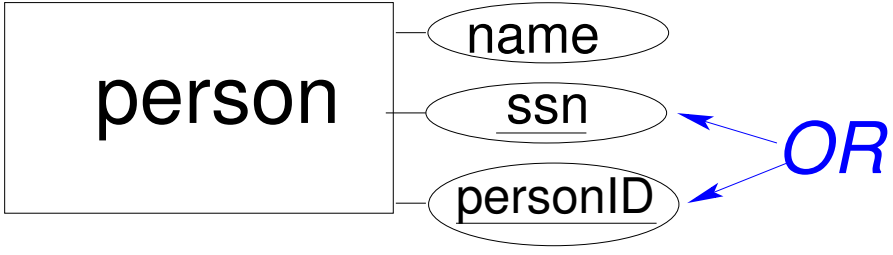
Multiple keys?

- Can an entity have multiple keys?

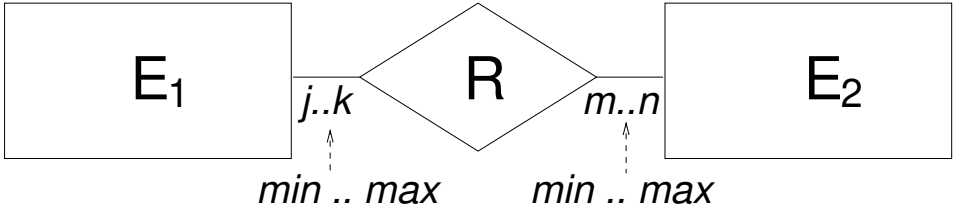
Multiple keys?

- Can an entity have multiple keys?

  Yes. Indicate using an *OR*.

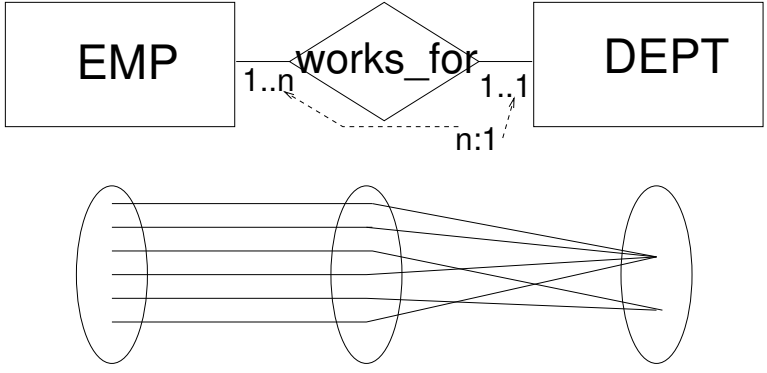


- *Binary Relationship* R on entities E_1 and E_2:

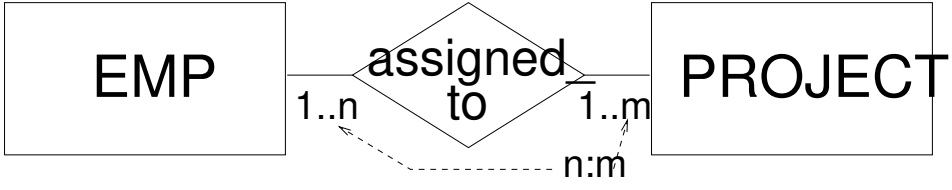


- Each instance of R is an association of an instance of E_1 with an instance of E_2.

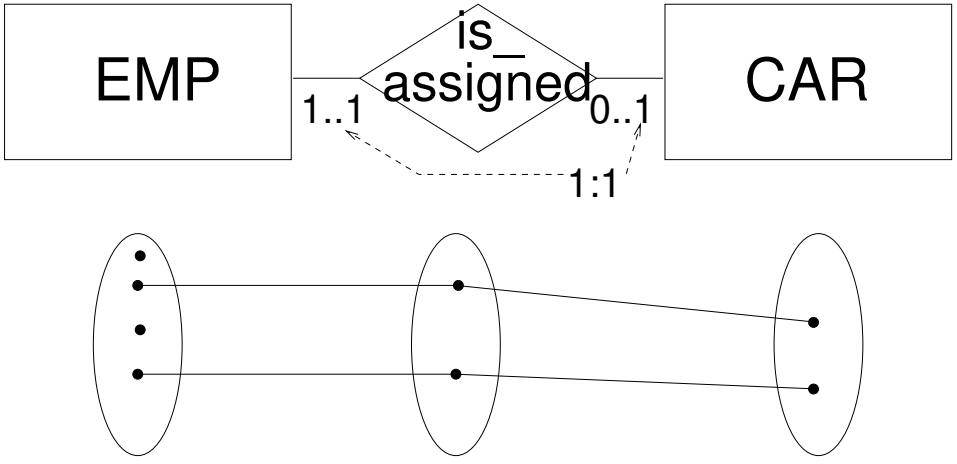
Cardinality Ratio 1 : N (a.k.a. N : 1)



Cardinality Ratio $N : M$



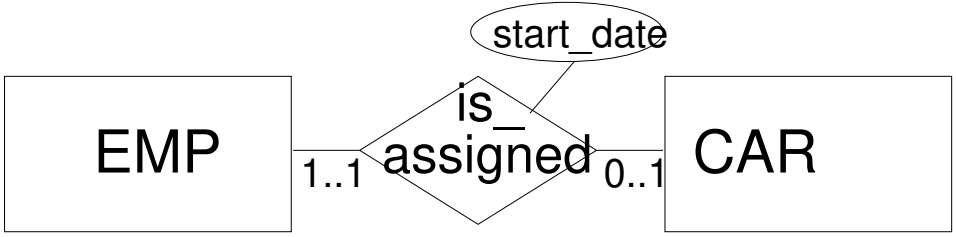
Cardinality Ratio 1 : 1



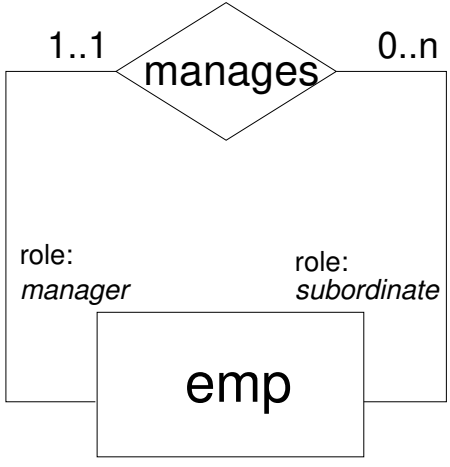
Type of Relationships

- What is the type of R ?
- An instance of R is a subset of $E_1 \times E_2$, i.e., a binary relation on E_1 and E_2.
- So, the type of R is ...

Relationships can have attributes too



Ring (Recursive) Relationships

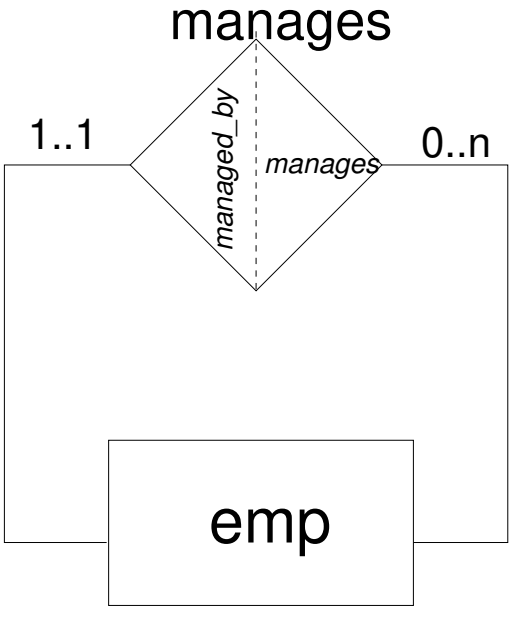


- Each instance emp plays two roles in the relationship manages.

  In the role of a manager, he/she manages 0..N others;

  In the role of a subordinate, he/she is managed by exactly one emp.

Ring (Recursive) Relationships



- This is the book's style.

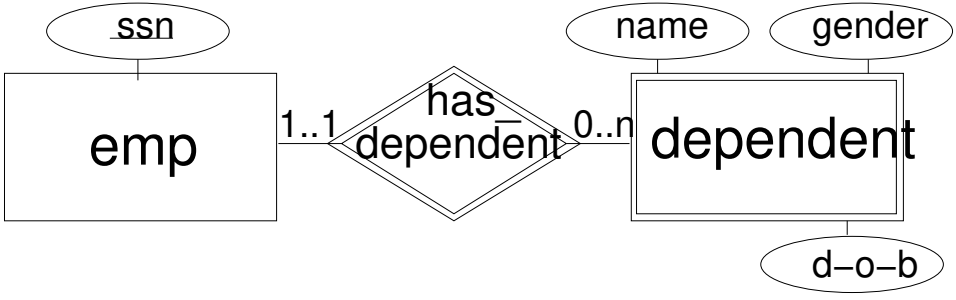
- *Total Participation* of entity E_1 in R:
  every instance of E_1 is associated with some instance of E_2.
  Also called *existence dependency*.
- *Partial Participation* of entity E_1 in R: *not* every instance of E_1 is associated with an instance of E_2.

- *Weak Entity* W is an entity with no key attribute.

  Instances of W are identified by borrowing the identifying attribute(s) of another entity O (the *Identifying Owner*) with which it has an (*Identifying*) relationship with total participation.

  There should be exactly one identifying owner. (Beware: Ignore Fig 3.20 in book.)

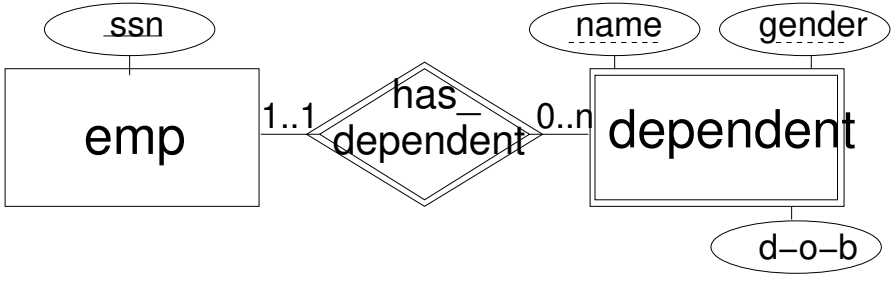
Weak Entity



- *dependent* is a weak entity;
  emp is its identifying owner; and
  has_dependent is the identifying relationship.
- Note the total participation of the weak entity in the identifying relationship.

Weak Entity with Partial Key

- *Partial Key* attribute(s) of weak entity: for a given owner, no two weak entity instances can have the same value of the partial key attribute(s).
- Use dotted underlines for partial key attributes.
- Partial key attributes + key attribute(s) borrowed from owner provides complete identification.



Weak Entity with No Partial Key

- What if there is no partial key?
- That means, for a given owner, no two instances of the weak entity can have the same value of *all* its attributes.
- i.e., *all* the attributes of the weak entity plus the borrowed attributes from its identifying owner provides identification.

Owner of a weak entity

- By contrast, entities that are *not* weak, i.e., the regular ones, i.e., the ones we started with, are sometimes called *strong* entities.
- A weak entity must have exactly one identifying relationship (denoted by a double-walled diamond) with an owner entity.

Owner of a weak entity

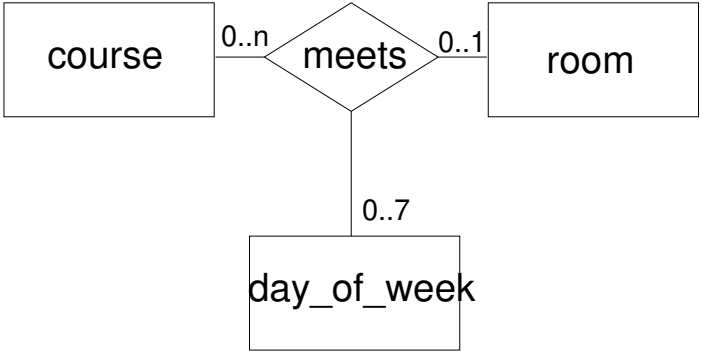
- Can the owner of a weak entity be another weak entity?
- Can a weak entity have non-identifying relationships as well?

Owner of a weak entity

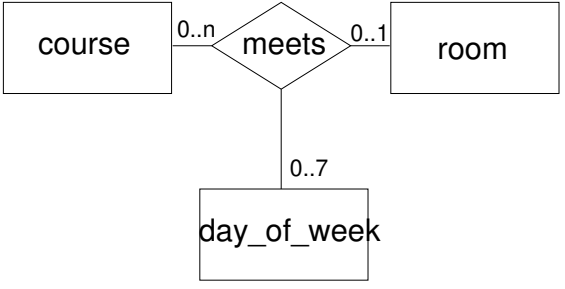
- The owner of a weak entity is either
  - a strong entity; or
  - another weak entity which has an identifying relationship with a strong entity; or
  - another weak entity which has an identifying relationship with a chain of identifying relationships culminating in a strong entity.
- A weak entity can have any number of non-identifying relationships (denoted by single-walled diamonds).

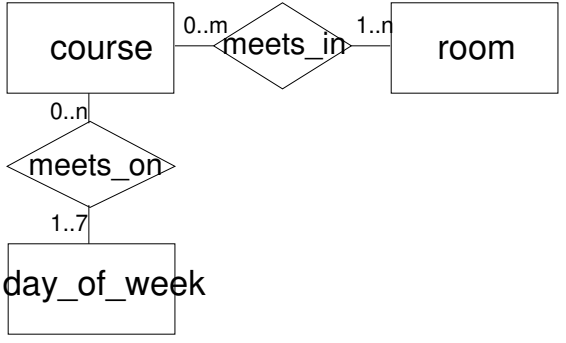
k-ary Relationship ($k \geq 2$)

- *k-ary relationship R on entities $E_1, E_2, ..., E_k$*: an instance is a subset of $E_1 \times E_2 \times ... \times E_k$, i.e., a k-ary relation on $E_1, \ldots, E_k$.
- *Degree* of $R = k$

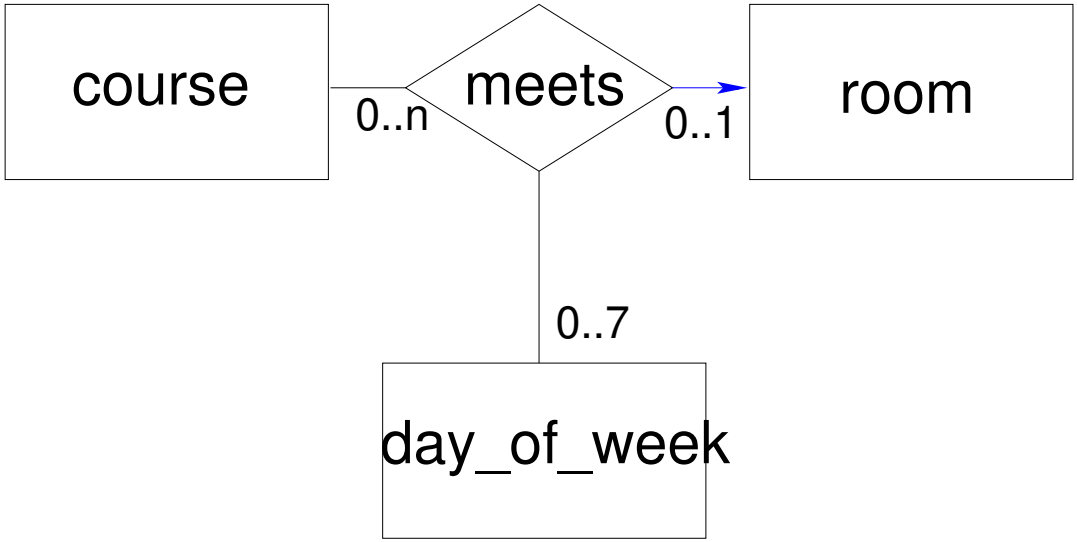


Are these two equivalent?





Arrow notation



- In a particular day of week, a particular course cannot meet in more than one room.

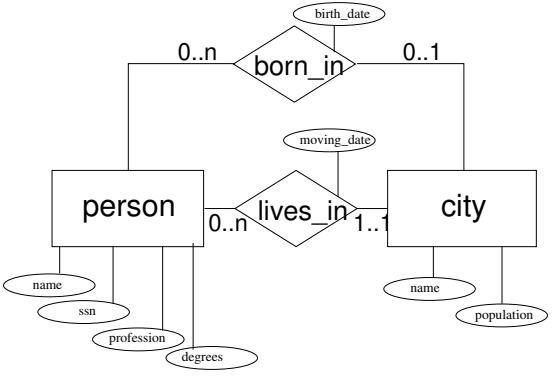
Arrow notation

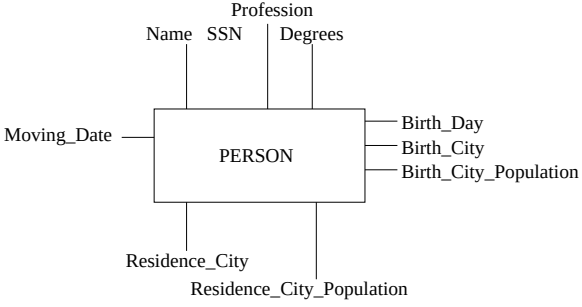
- Intuitively...

Suppose
c_i is an instance of Course,
d_j is an instance of day_of_week, and
r_k, r_m are instances of room.

Scan any instance of meets (it's a set of triples).
If you find (c_i, d_j, r_k),
then you will never find (c_i, d_j, r_m), where $k \neq m$,

Are these two equivalent?





- Acme Inc. is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

An Example: *Acme Inc.*

- We store each employee's name, social security number (unique), address, salary, gender, and birthdate. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
- We want to keep track of dependents for insurance purposes by storing their name, gender, birthdate, and relationship to the employee.

Pitfalls of Natural Language Descriptions

- Requirements of an organization are expressed in natural language and may be the first input to the designer.

Requirements of a University Database

In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, places and states where they lived before (with the period they lived there), courses that they have passed, with name, code, professor, grade, and date.

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate students, we represent the name of the adviser and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name, age, place, and state of birth, name of the department they belong to, telephone number, title, and topics of their research.

Red Flags

In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, places and states where they lived before (with the period they lived there), courses that they have passed, with name, code, professor, grade, and date.

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate students, we represent the name of the adviser and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name, age, place and state of birth, name of the department they belong to, telephone number, title, and topics of their research.

Changes

In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, places and states where they lived before

Too Abstract: replace by *cities*

Changes

In a university database, we represent data about students and professors. For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, places and states where they lived before (with the period they lived there), courses that they have

Too Abstract: replace by *number of years*

Changes

We also represent courses they are presently *attending,*

Too Abstract: replace by *in the current year*

Changes

We also represent courses they are presently attending, and for each day, places and hours where classes are held

day: implicit reference to *week;*
replace by
day of the week

Changes

We also represent courses they are presently attending, and for each day, places and hours where classes are held

places: homonym for *places* earlier: replace by *rooms*

Changes

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate

classes: synonym for *courses* earlier: replace by *courses*

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate students, we represent the name of the adviser and the

adviser: implicit reference to *professor*;
replace by
professor who is the adviser

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate students, we represent the name of the adviser and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name,

teacher: synonym for earlier *professor*; replace by *professor*

We also represent courses they are presently attending, and for each day, places and hours where classes are held (each course meets at most once a day). For graduate students, we represent the name of the adviser and the total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name, age, place and state of birth, name of the department

we have seen this before ... replace by *cities*

total number of credits in the last year. For Ph.D. students, we represent the title and the research area of their thesis. For teachers, we represent last name, age, place and state of birth, name of the department they belong to, telephone number, title, and topics of their research.

synonym for earlier *research area*; replace by research area.

- Look for terms that are either too abstract or too specific. Choose an appropriate level of abstraction.

- Attack circumlocution.

  Every department has a secretary who has another important function: to greet visitors; he/she is called a receptionist.

  versus

  Every department has a secretary who is also a receptionist.

- Uniformity in style is helpful.

- Unravel hidden synonyms and homonyms.
- Make implicit references explicit.
- A glossary of terms could be useful.

Rewritten Requirements

In a university database, we represent data about students and professors.

For students, we represent last name, age, sex, city and state of birth, city and state of residence of their families, cities and states where they lived before (with the number of years they lived there), courses that they have passed, with name, code, professor, grade, and date.

Rewritten Requirements

We also represent courses they are attending in the current year, and for each day of the week, rooms and hours where courses are held (taught) (each course meets at most once a day).

For graduate students, we represent the name of the professor who is the adviser and the total number of credits in the last year.

Rewritten Requirements

For Ph.D. students, we represent the title and the research area of their thesis.

For professors, we represent last name, age, city and state of birth, name of the department they belong to, telephone number, title, and research areas.