Part 1: ER-Diagrams: Repetition and Extension

References:

- Barker: CASE*Method, Entity Relationship Modelling.
  Addison-Wesley, 1990, ISBN 0-201-41696-4, ca. $61.
- Koletzke/Dorsey: Oracle Designer Handbook, 2nd Edition.
  ORACLE Press, 1998, ISBN 0-07-882417-6, ca. $40.
- A. Lulushi: Inside Oracle Designer/2000.
  Prentice Hall, 1998, ISBN 0-13-849753-2, ca. $50.
- Oracle/Martin Wykes: Designer/2000, Release 2.1.1, Tutorial.
  Part No. Z23274-02, Oracle, 1998.
- Heli Helskyaho: Oracle SQL Developer Data Modeler for Database Design Mastery.
  McGraw Hill Education / Oracle Press, 2015, ISBN 0071850090, 336 pages.
- Teorey: Database Modeling & Design, 3rd Edition.
  Morgan Kaufmann, 1999, ISBN 1-55860-500-2, ca. $32.
- Elmasri/Navathe: Fundamentals of Database Systems, 2nd Ed.,
  Appendix A, "Alternative Diagrammatic Notations".
- Rauh/Stickel: Konzeptuelle Datenmodellierung (in German), Teubner, 1997.

Objectives

After completing this chapter, you should be able to:

- enumerate the ER-constructs supported by Oracle Designer.
- draw ER-diagrams in the graphical syntax of Oracle Designer ("Barker Notation").
- You should also be able to read such diagrams.
- explain the difference between the global DB schema and the views contained in single diagrams.

Overview

Entity-Relationship-Model (1)

- The Entity-Relationship-Model is called a "semantic data model", because it more closely resembles the real world than e.g. the relational model.
  - ◇ In the ER-model, persons are modelled. In the relational model, only their names/numbers.
  - ◇ In the ER-model, there is a distinction between entities and relationships. In the relational model, both are represented by relations.

    This expressiveness is not needed to satisfy the information requirements of the applications. But it makes the correspondence between the schema and the real world clearer (like a comment).

Entity-Relationship-Model (2)

- Proposed by Peter Pin-Shan Chen (1976).
- There is a useful graphical notation which helps to establish a better overview; to "see" the structure of the data.

  The graphical notation also helps to communicate with the future users.
- Many variations and extensions of the ER-Model have been proposed.

  We use here the notation proposed by Richard Barker (his book appeared 1989/90). The easy extensibility of the ER-model (because it is not bound to a DBMS) is also one of the strengths of the model.

Entity-Relationship-Model (3)

- There are specialized graphical editors and other design tools.

  E.g. Oracle Designer and Oracle SQL Developer Data Modeler are CASE tools for DB-applications, and one component of them is a graphical editor for ER-diagrams. (CASE: Computer-Aided Software Engineering.) Alternatives: Sybase PowerDesigner, CA ERwin, ...

- The ER-model is a standard tool for conceptual DB design. However, recently, object-oriented methods are also used.

  In Software Engineering, UML (the Unified Modeling Language) has become a standard. One can use UML class diagrams for DB design. They are based on the ER-model, but they have no keys (only as an extension). Keys a a central database concept.

Basic ER-Model Concepts (1)

Entities:

- Objects in the miniworld about which information has to be stored. E.g. persons, books, courses.

  It does not matter whether entities have a physical existence (and can be touched) or only a conceptual existence.

- At each instant, the miniworld that has to be modelled can contain only a finite number of entities.

  E.g. "all numbers" (infinitely many) cannot be entities.

- It must be possible to distinguish entities from each other, i.e. they must have some identity.

  So ants in a heap do not qualify as entities.

Basic ER-Model Concepts (2)

Data Type Elements:

- Values from some possibly infinite set, which can be stored and printed.

  E.g. strings, numbers, dates, lengths, pictures. A person cannot be stored (entity), but his/her name can be stored (data type element).

- Most current DBMS have some predefined set of data types which they support.
- It is possible to use non-standard types in ER-schemas (complicates the later logical design).

  The ER-Design should be independent of a specific DBMS.

Basic ER-Model Concepts (3)

Attribute:

- A property or characteristic of an entity.

  Depending on the specific version of the ER-model, also relationships can have attributes (see below).

- E.g. the title of this course is "Database Design".
- The value of an attribute is an element of a data type like string, integer, date: It has a printable representation.
- Attributes can be optional, i.e. permit null values.

  Their semantics is then a partial function from entities to data type values.

Basic ER-Model Concepts (4)

Relationship:

- Relation between pairs of entities ("binary relationship").

  Some ER-notations allow relationships involving more than two entities. My experience shows that this often leads to errors.

- E.g. I (a person) teach "Database Design" (a course).

- The word "Relationship" is also used as an abbreviation for "Relationship-Type" (see below).

  It should be clear from the context what is meant.

Basic ER-Model Concepts (5)

Entity-Type:

- Set of similar entities (with respect to the information which has to be stored about them), i.e. entities which have the same attributes.
- E.g. all faculty members of this university.

Relationship-Type:

- Set of similar relationships.
- E.g. "X teaches course Y".

ER-Model Variants

- Variants of the ER-Model differ in:
  - ◇ The selection of ER modeling constructs.

    See next page for the ER constructs supported in Oracle Designer.
  - ◇ The notation used for these constructs.

    E.g. softboxes are used for entities, and the "crowsfoot" / "chicken feet" notation for cardinalities.
  - ◇ The possibility to model also behaviour: Methods/Operations supported by the entities.

    This is typical for object-oriented approaches. Oracle Designer has other tools for modeling this (Process Diagrams, Dataflow Diagrams).

Supported ER-Constructs (1)

Oracle Designer supports the following ER-constructs:

- Binary relationships including recursive ones.
- The most important cardinalities.

  0 and 1 as minimum, 1 and $*$ as maximum. The cardinality $(1,*)$ on both sides of the relationship is excluded (insertion would be too difficult). In the repository, arbitrary min-max cardinalities can be specified, but they are not shown in the diagrams.
- Optional attributes (null values).
- Domains for attributes.

  Domains are similar to user-defined data types, however, they are only names for the standard datatypes of the DBMS. See below.

Supported ER-Constructs (2)

- Constraints on attribute values.
- Keys of entity types.
- Weak entities.

  I.e. using a relationship as part of the key of entities.
- Disjoint and total specialization.
- Mutually exclusive relationships.
- Non-transferable relationships.
- Various additional information about entities.

  E.g. synonyms, expected sizes, comments, further documentation.

Supported ER-Constructs (3)

Oracle Designer does not support these ER-constructs:

- Ternary etc. relationships.

  As explained below, one can always replace relationships by "association entities" and binary relationships.

- Relationship attributes.

  Also in this case, one must turn the relationship into an (association) entity with two binary relationships without attributes.

- Multivalued/structured attributes.

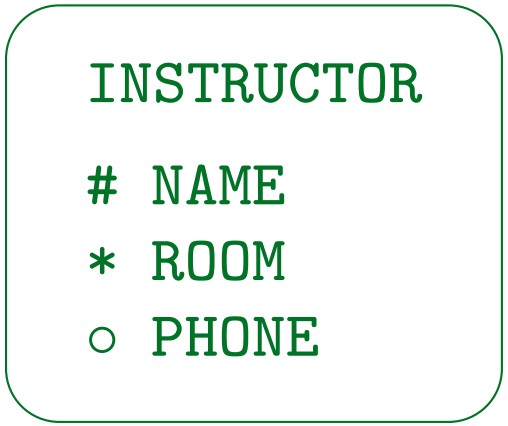
  Multivalued attributes can be handled with weak entities. For structured attributes, the components can be declared as attributes.

- Non-disjoint specialization.

Overview

Entities and Attributes (1)

- The Oracle tool uses a "softbox" (rectangle with rounded corners) for entity types.
- Attribute names are written into this box:



- Attributes which are mandatory (not null) are marked with *, optional attributes with ○.

Entities and Attributes (2)

- Key attributes are marked with #.

  Attributes marked with # together constitute the primary key.
  Key attributes are automatically mandatory, no * is needed.

- Oracle Designer allows to customize what is displayed, e.g. it is possible not to show attributes.

  This is useful e.g. in order to get an overview of a large schema.

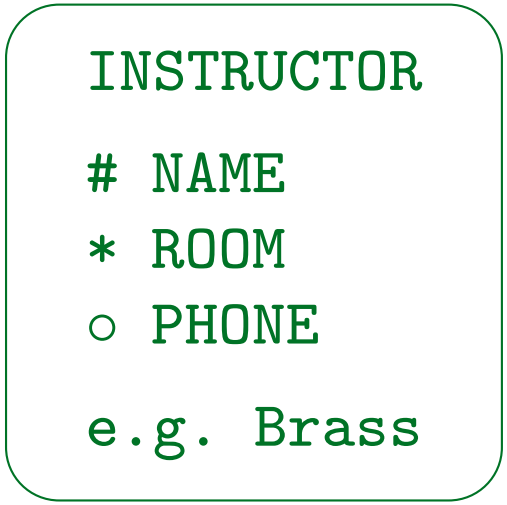
- More information is stored about entities which is not graphically displayed.

  One can open a dialog box with additional entity properties.

- Entity type names must be unique in the schema.

Entities and Attributes (3)

- In earlier versions, one could mention an example entity at the bottom of the entity type box:

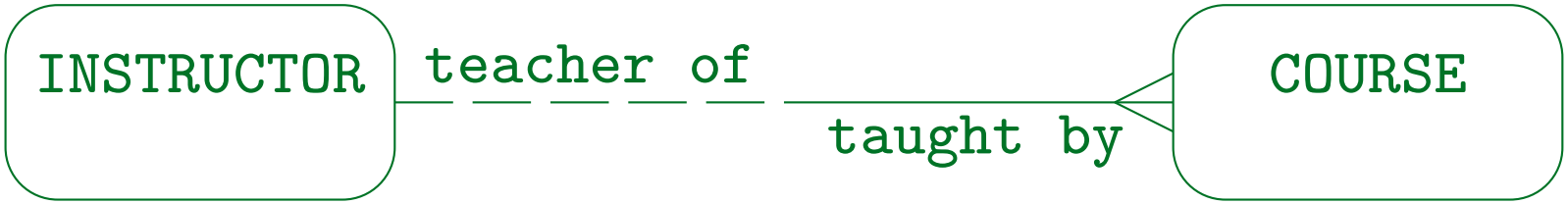


- This is a nice idea, but it is no longer supported.

  Of course, one can and should mention an example in the description of the entity type (not shown in the diagram).

Relationships (1)

- Relationships are marked by lines between the entity boxes (no diamond).
- The form of the line (dashed or solid) and the line end (simple or crowsfoot) describe the cardinalities:



- This is very illustrative: One instructor can teach many courses, but each course is only taught by one instructor (see below).

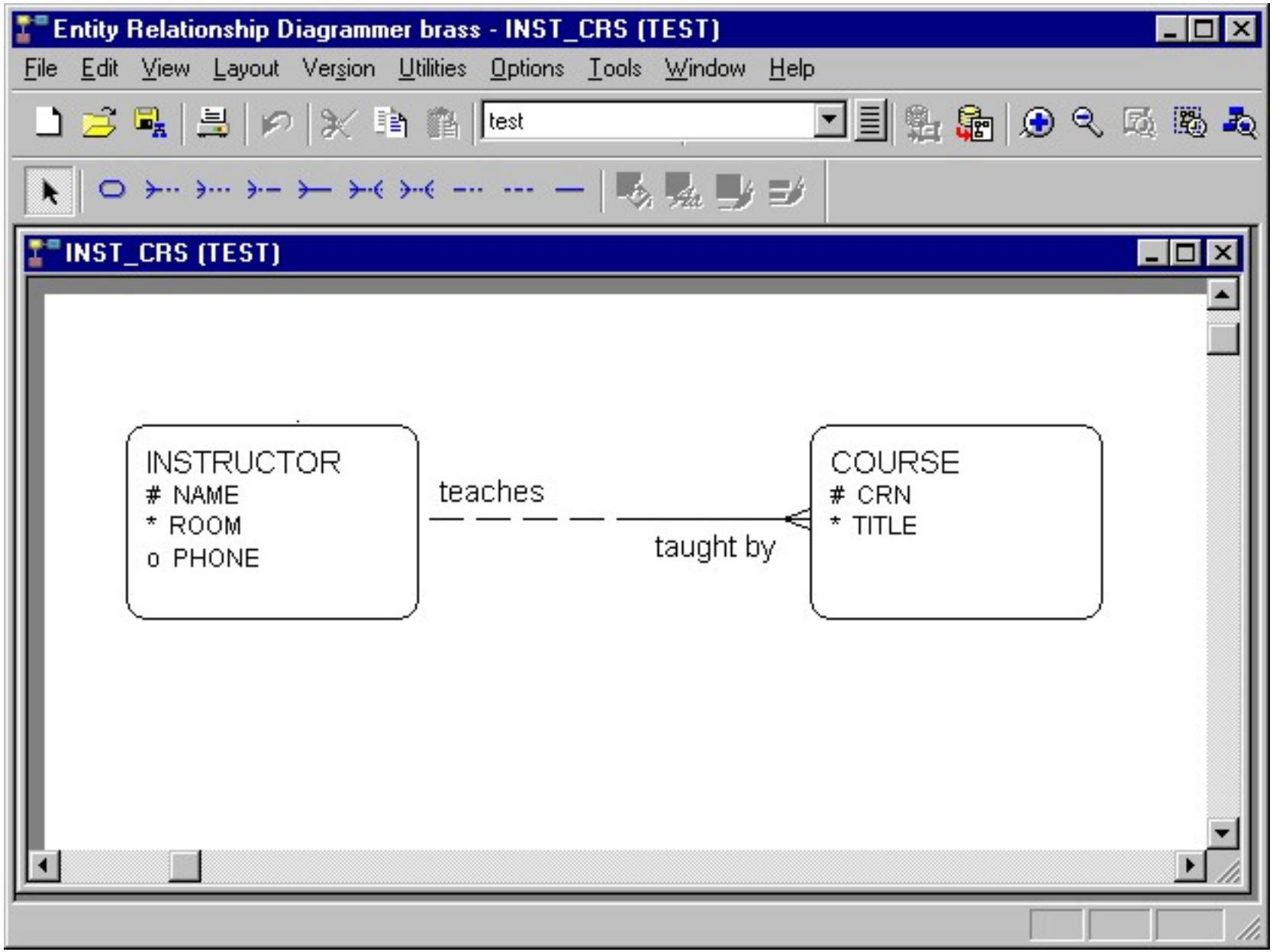
Relationships (2)

- Relationships have two names (seen from each of the ends): The "from name" and the "to name".

  The relationship names correspond to the role names of entities used in the "diamond" notation at least for recursive relationships.

- Just as a relationship between people, relationships are "both way": This is reflected in the two names.
- The same names can be used for different relationships (they do not have to be globally unique).

  Only between the same two entity types there cannot be two relationships that agree in both, to and from name.


Entity Relationship Diagrammer brass - INST_CRS (TEST)
File Edit View Layout Version Utilities Options Tools Window Help
test
INST_CRS (TEST)
INSTRUCTOR
NAME
* ROOM
o PHONE
teaches
taught by
COURSE
CRN
* TITLE

Notation for Cardinalities (1)

- In the first DB course, the (min,max)-notation for cardinatities was introduced:

- An instructor entity can be related to any number of course entities (between 0 and arbitrarily many).
- A course entity must be related to exactly one instructor entity (minimally 1 and maximally 1).

Notation for Cardinalities (2)

- As maximum cardinalities, only `1` and `*` are common. The maximum cardinalities on both sides classify a relationship as
  - ◇ Many-to-many (N:M): "`*`" on both sides.
  - ◇ One-to-many (1:N): "`*`" and "`1`".
  - ◇ One-to-one (1:1): "`1`" on both sides.
- The example is "one-to-many" from instructor to course, i.e. one instructor can teach many courses.

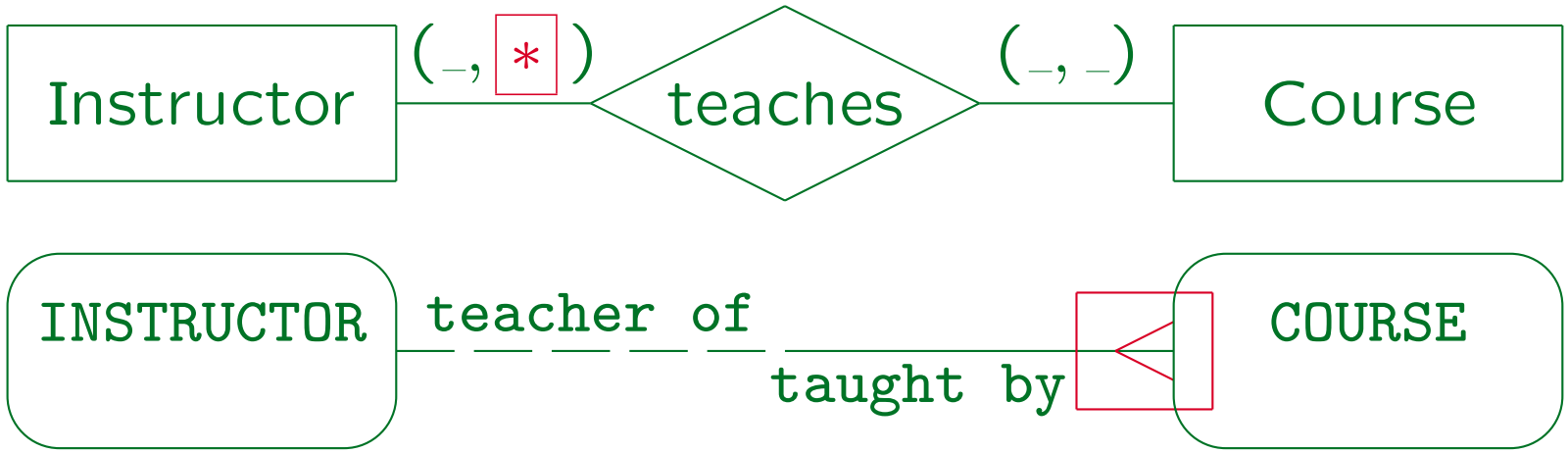
  The maximum cardinality "`*`" is written on the instructor side (i.e. the "one" side): The (min,max)-cardinalities on the instructor side describe the outgoing edges from a single instructor (number of courses).

Notation for Cardinalities (3)

- As minimum cardinalities, only 0 and 1 are common:
  - ◇ The minimum cardinality "0" means optional (or partial) participation in the relationship:
    Not every instructor must teach a course.
  - ◇ The minimum cardinality "1" means mandatory (or total) participation in the relationship:
    Every course must be taught by an instructor.
- A relationship can be optional on both sides, optional on one side and mandatory on the other, or mandatory on both sides (difficult for insertions).

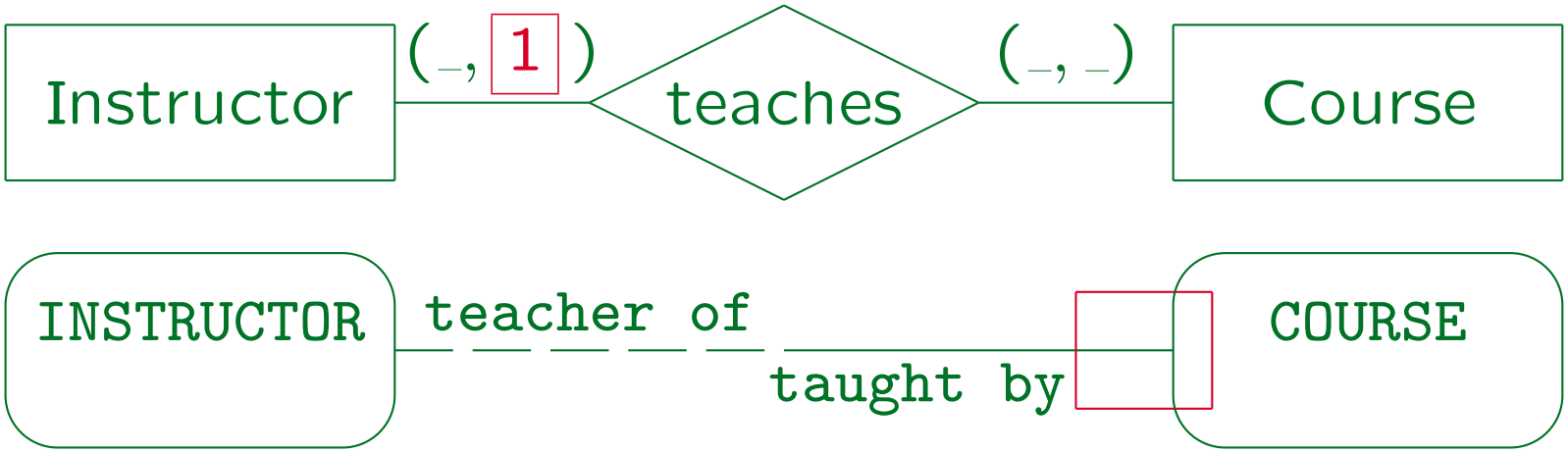
Notation for Cardinalities (4)

- The notation used in Oracle Designer can represent the common maximum cardinalities (1 and $*$).
- For the maximum cardinality "$*$" on the instructor side, a crowsfoot is drawn on the course side:



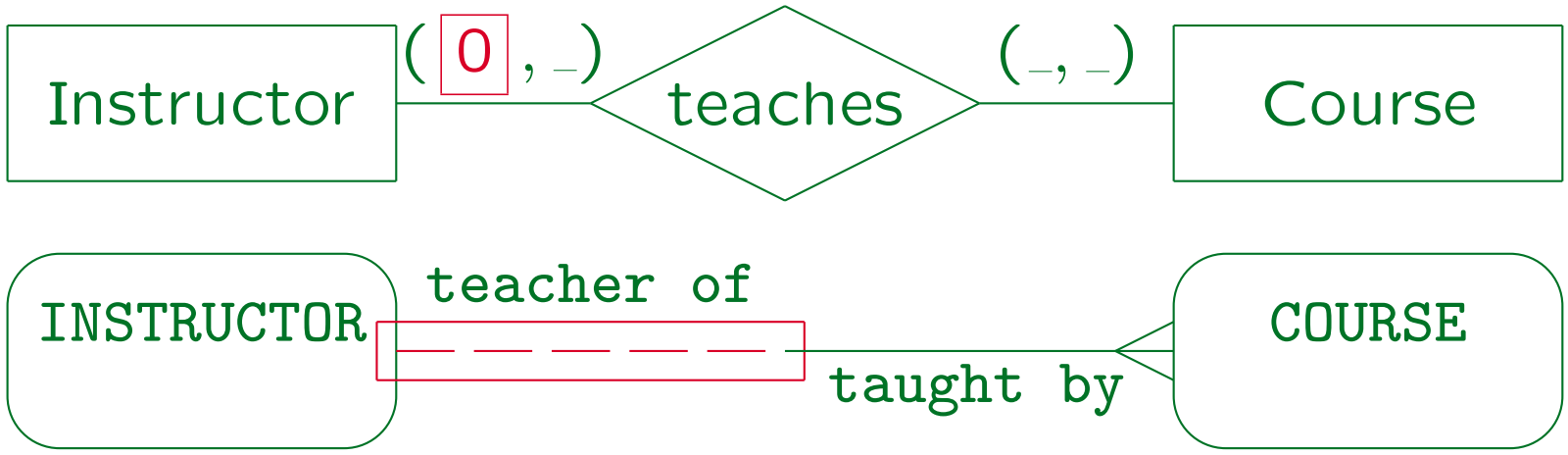
Notation for Cardinalities (5)

- If the maximum cardinality should be "1" on the instructor side (each instructor can teach only one course), no crowsfoot is drawn on the course side:



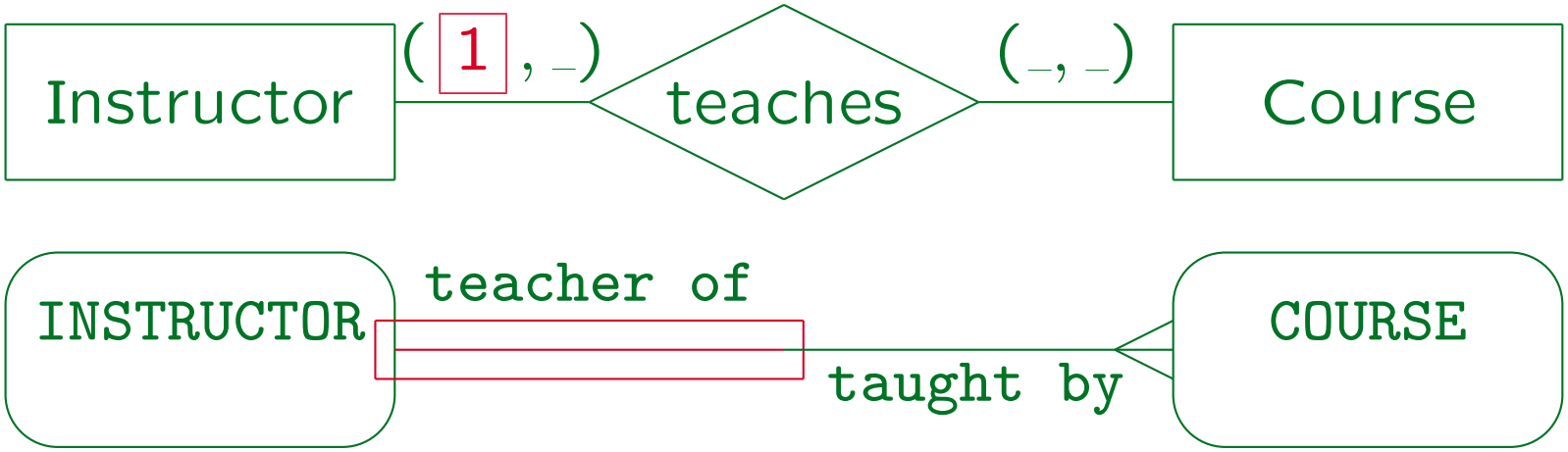
Notation for Cardinalities (6)

- The notation used in Oracle Designer can represent the common minimum cardinalities (0 and 1).
- For the minimum cardinality "0" on the instructor side, a dashed line is drawn on the instructor side:



Notation for Cardinalities (7)

- For the minimum cardinality "1" on the instructor side (each instructor must teach at least one course), a solid line is drawn on the instructor side:



Checking Cardinalities (1)

- If the role names of the relationship ("teacher of", "taught by") are chosen rigorously, natural language sentences that explain the cardinalities can be automatically generated.
  - ◇ "Each (and every) INSTRUCTOR may be teacher of one or more COURSES."
  - ◇ "Each (and every) COURSE must be taught by one and only one INSTRUCTOR (ever)."

    Phrases in parentheses only emphasize, but don't change the meaning. They can be left out.

Checking Cardinalities (2)

- "May be" indicates optional participation, "must be" is used for mandatory participation.

  Oracle Designer knows the plural form of every entity type, as required for generating these sentences. Some design reports that the "Repository Reports" utility produces contain such sentences.

- Note that both sentences are needed to completely describe the relationship.
- However, it is sometimes difficult to choose relationship names that fit into this pattern.

  They must consist of a noun (role) and a preposition. For verbs like "teaches" a slightly different pattern would be needed.

Checking Cardinalities (3)

- Such sentences can be shown to domain experts (future users) who cannot read ER-diagrams.
- This is a way of validating the database schema (checking it for correctness).

  Keys and other constraints could be validated in the same way.
- Actually, one can even produce a question form and let the user check whether it is true:

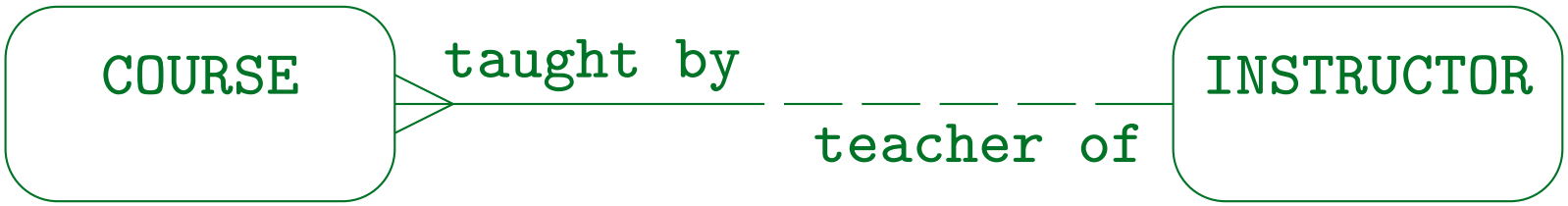
  Each and every course must be taught by one and only one instructor, is that true?

  ☐ yes ☐ no

Cardinalities (1)

- The toolbar has nine different relationship types.
  The first is "many to one (mandatory to optional)":

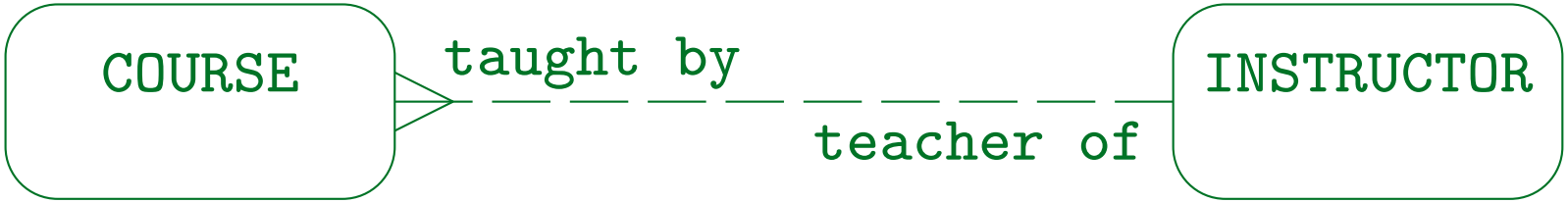
- In Oracle Designer:



Cardinalities (2)

- The next is "many to one (optional to optional)":

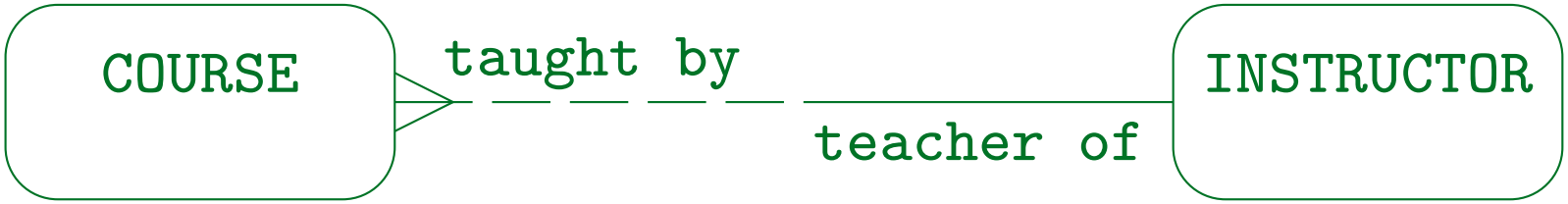
- In this case, a course has not necessarily a teacher assigned.
- In Oracle Designer:



Cardinalities (3)

- Many to one (optional to mandatory):

- In Oracle Designer:



Here an instructor must teach at least one course, and can teach any number of courses. A course does not require an instructor, but can have at most one.

Cardinalities (4)

- Many to one (mandatory to mandatory):

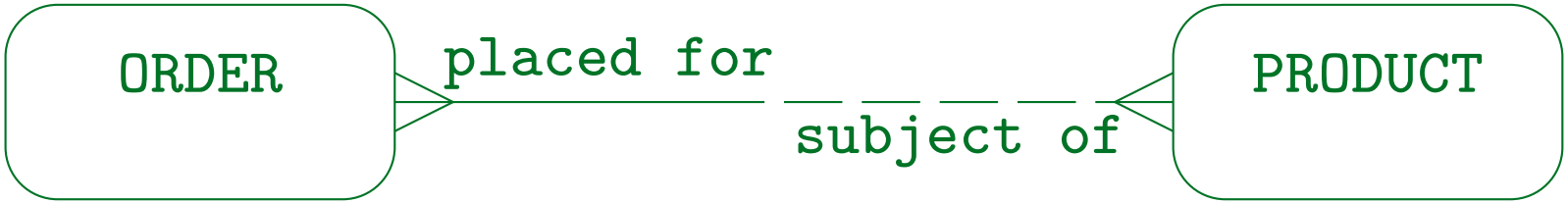
- In Oracle Designer:

Every invoice item belongs to exactly one invoice. An invoice can consist of several items, but must consist of at least one.

Cardinalities (5)

- Many to many (mandatory to optional):

- In Oracle Designer:

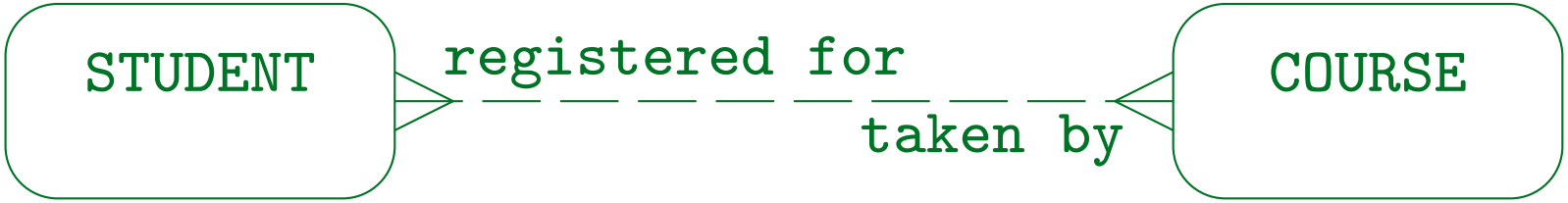


Every purchase order must be for at least one product, but can be for many products. One product can be ordered in many purchase oders. There can be new products that are not yet ordered.

Cardinalities (6)

- Many to many (optional to optional):

- In Oracle Designer:



- This is the most general relationship:

  A student can take any number of courses (including zero), a course can be taken by any number of students (again including zero).

Cardinalities (7)

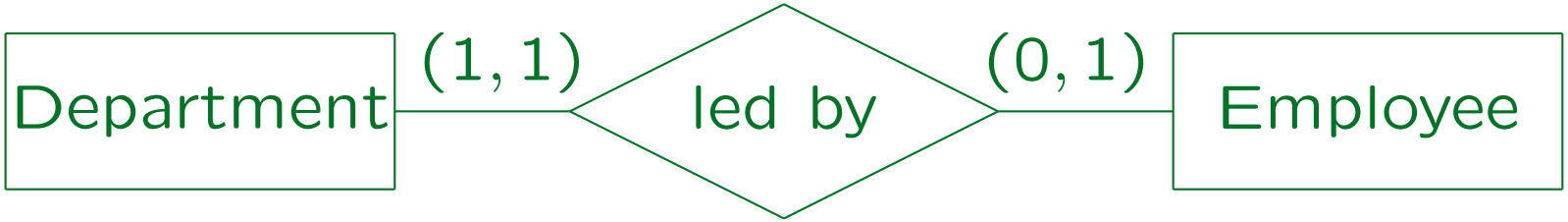
- Many to many (mandatory to mandatory):

- This is not supported in Oracle Designer.
- It would be very difficult to insert entities.

  A student cannot be inserted without a course, and a course cannot be inserted without a student. In general, when one defines cardinalities, one should think about elementary transactions. Which insertions/deletions must happen together such that the cardinality requirements are satisfied at the end of the transaction? If the transaction is too complicated, the cardinality requirements should be relaxed.

Cardinalities (8)

- One to one (mandatory to optional):



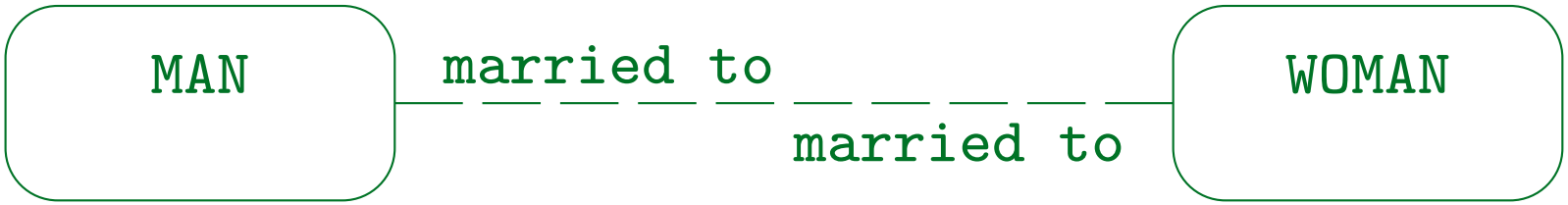
- In Oracle Designer:

Every department is led by exactly one employee, an employee can be head of at most one department.

Cardinalities (9)

- One to one (optional to optional):

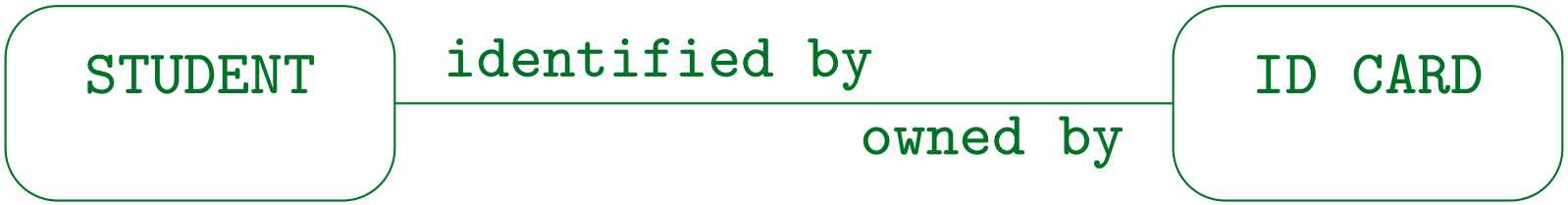
- In Oracle Designer:



Cardinalities (10)

- One to one (mandatory to mandatory):

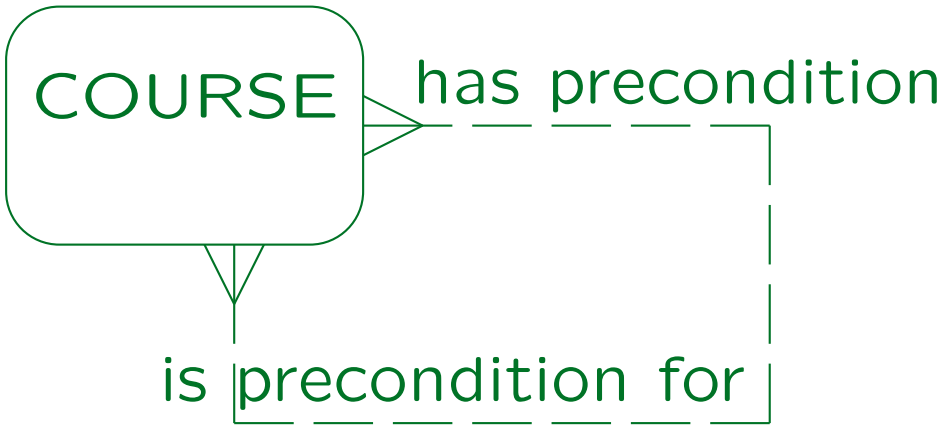
- In Oracle Designer:



- Uncommon. Consider merging the two entities.

Recursive Relationships

- Oracle Designer does support recursive relationships (between two entities of the same type).



- Oracle Designer actually displays recursive relationships with a three-quarter circle ("swine ear").

Overview

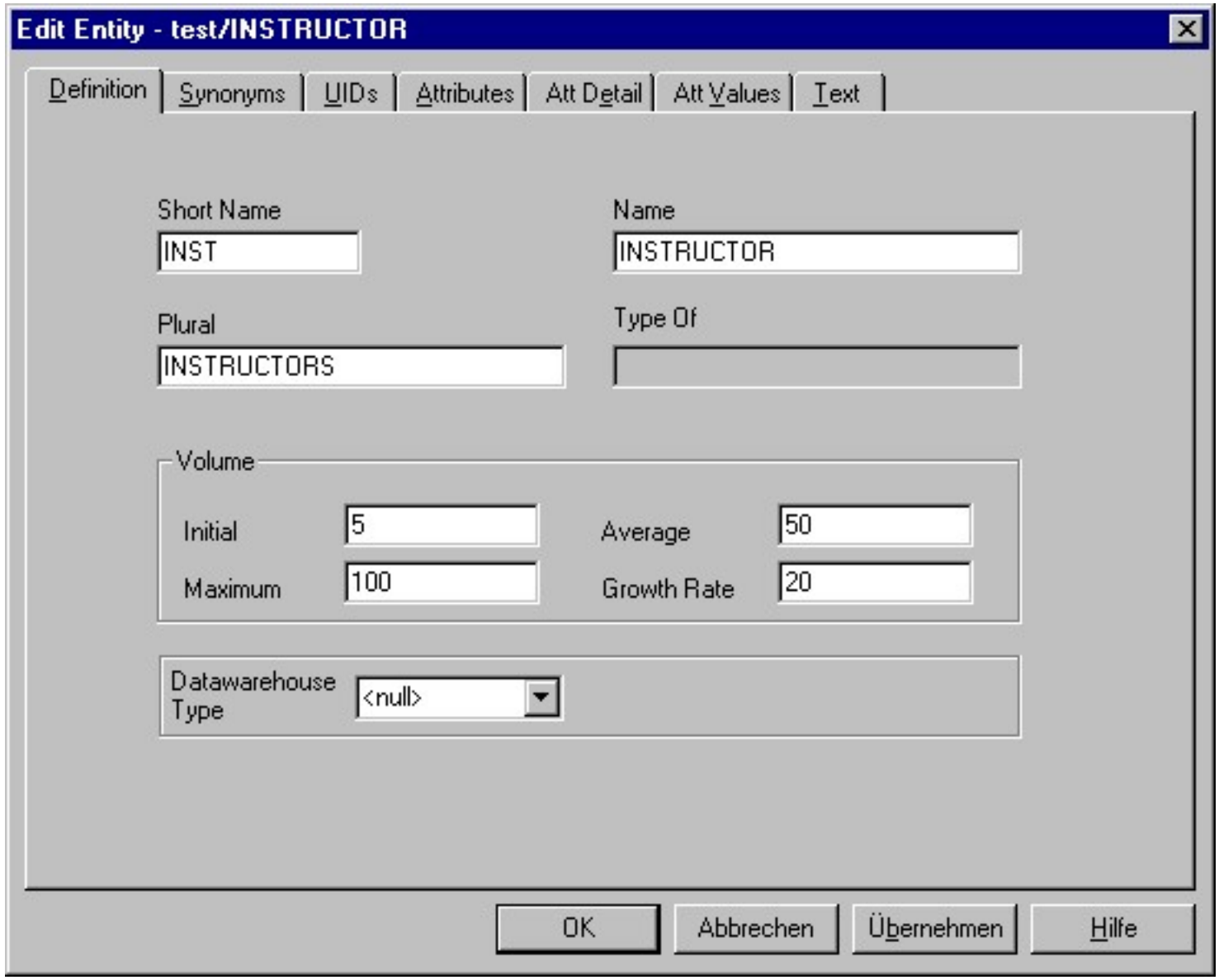
Entity Properties (1)

- By double clicking on an entity in a diagram, one opens the "Edit Entity" dialog box.
- It gives access to the properties of the entity, its attributes (including constraints for attribute values), unique identifiers (keys), and synonyms.
- In this way, much more information can be stored about the entity type than what is actually shown on the diagram.

  It is possible to customize what is shown in the diagram, e.g. all attributes, only mandatory attributes, only the primary key attributes, or no attributes.

Entity Properties (2)

- The dialog box has several tabs/pages:
  - Definition (name, plural, short name, etc.)
  - Synonyms (alternative names)
  - UIDs (keys and weak entity identification)
  - Attributes (list of all attributes)
  - Attribute Detail (one page for each attribute)
  - Attribute Values (constraints)
  - Text (documentation, notes).


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Short Name
INST
Name
INSTRUCTOR
Plural
INSTRUCTORS
Type Of
Volume
Initial
5
Average
50
Maximum
100
Growth Rate
20
Datawarehouse Type
<null>
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (4)

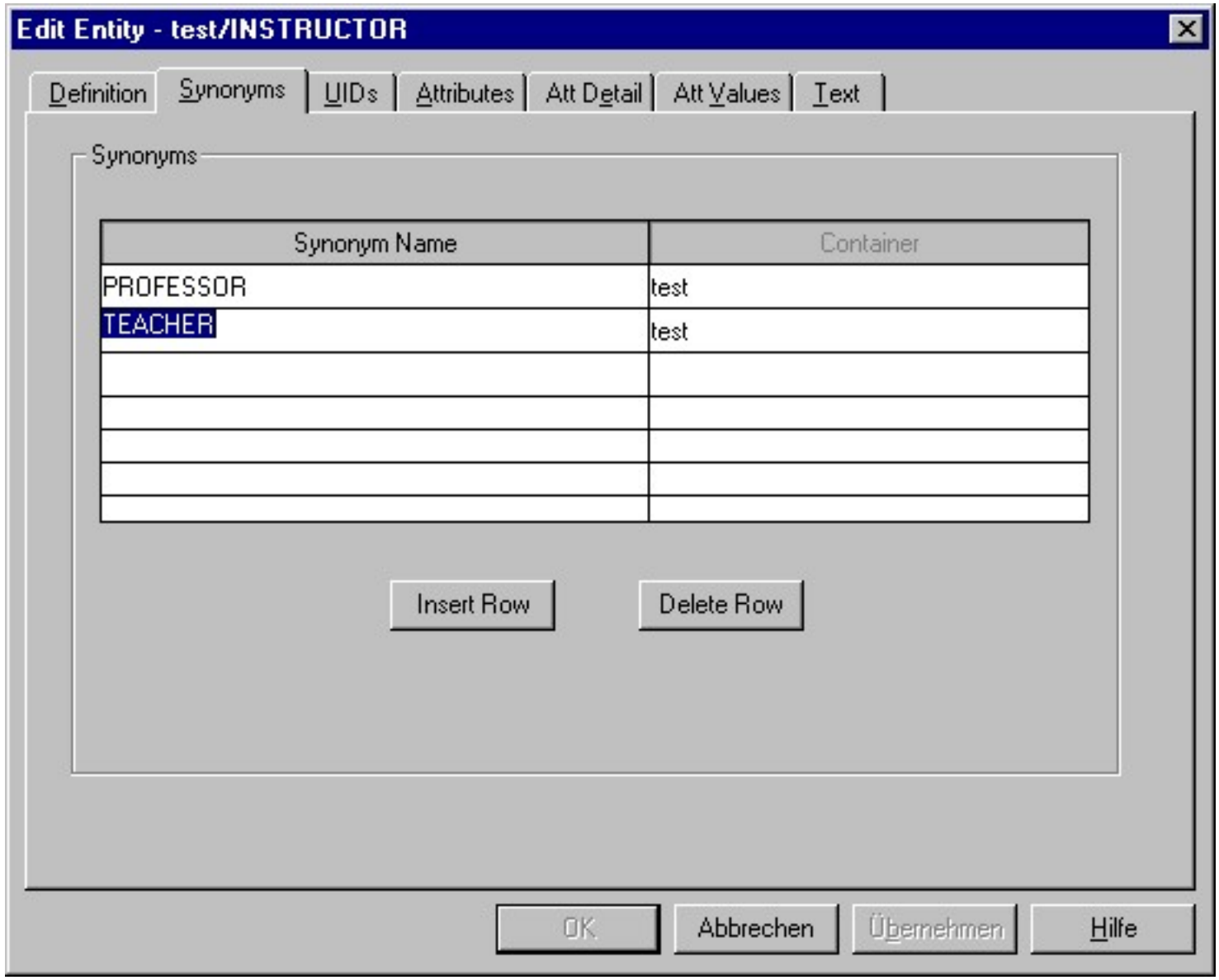
"Definition" Page:

- Names of an entity (Short name, Name, Plural).
- Super class: "type of" (if this is a subclass).
- Expected number of entities of this type.

  This information is important for physical design.
  The initial, average, and maximum volume (number of entities) can be specified, as well as the annual growth rate (in percent). The meaning is a bit unclear, e.g. whether maximum is increased by the annual growth rate, and over which time interval the average is taken.

- Datawarehouse type (if DW application).

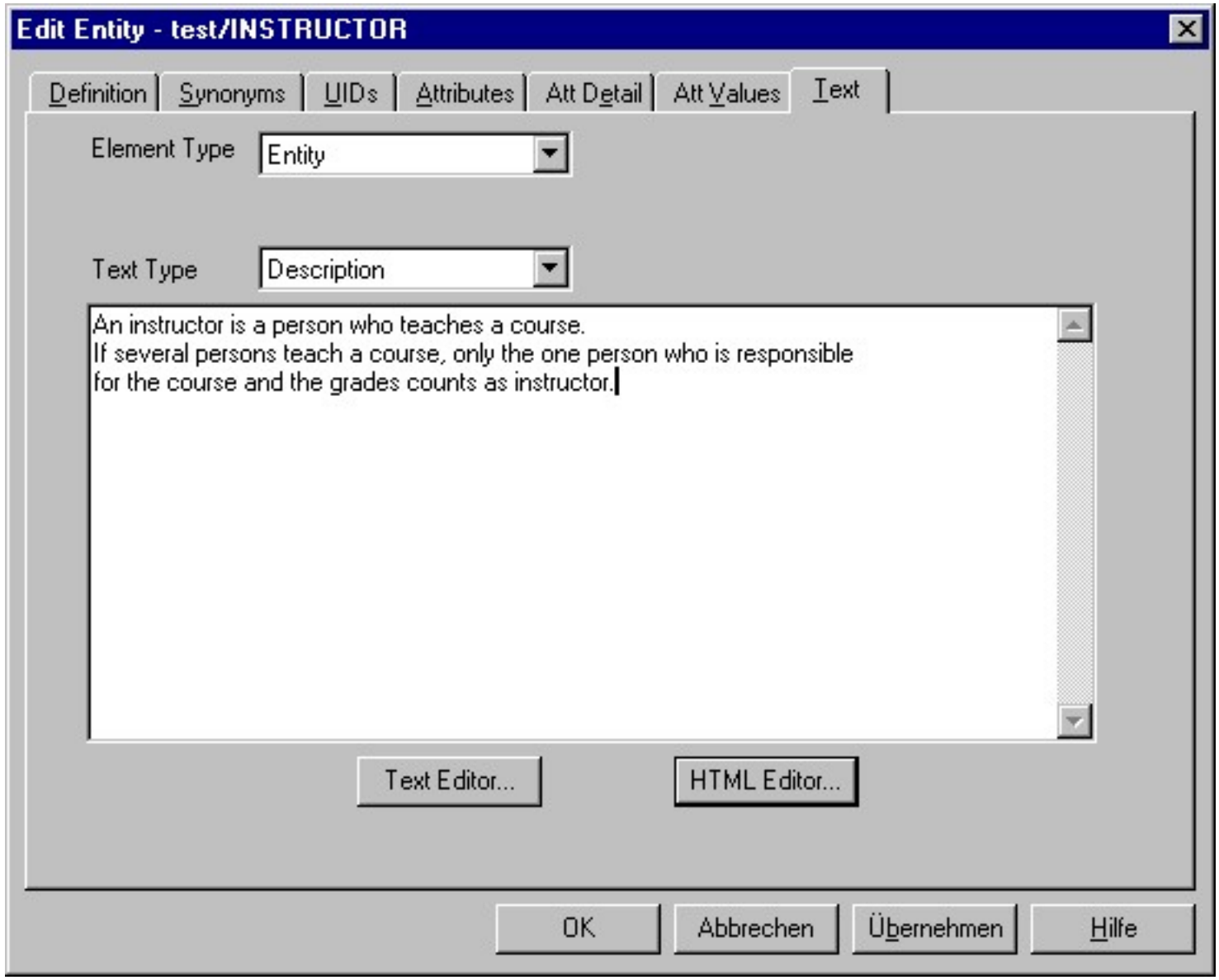
  "Fact" vs. "Dimension" tables (see below).


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Synonyms
Synonym Name
Container
PROFESSOR
test
TEACHER
test
Insert Row
Delete Row
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (6)

"Synonyms" Page:

- Alternative names for the entity can be defined.
- It is an important task in database design to check whether things named differently by different users are really the same concept.
- The Oracle DBMS permits to define synonyms for tables and other database objects (not in SQL-92).
- However, it is also possible to treat synonyms only as part of the design documentation, and not to reflect them in the final relational schema.

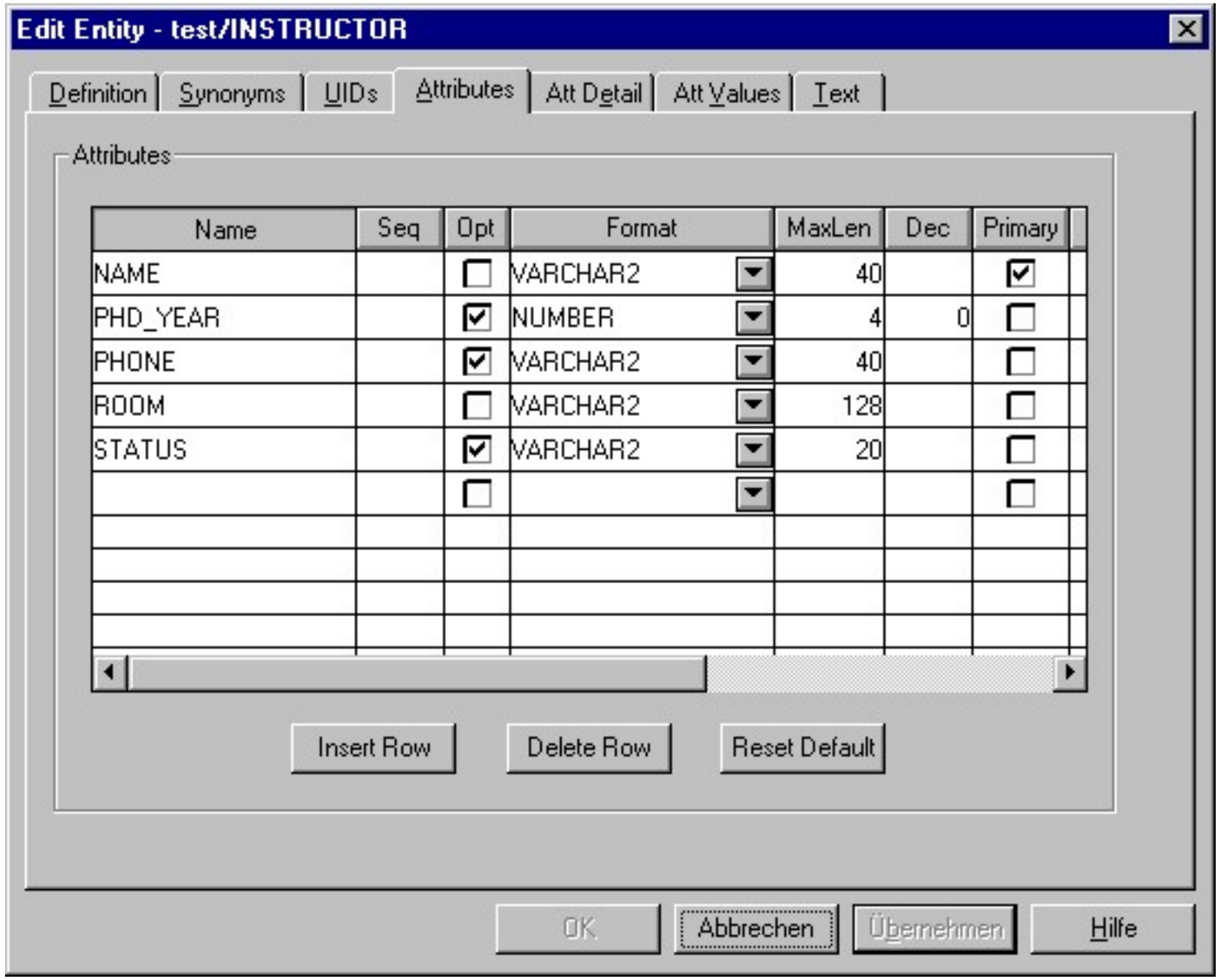

Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Element Type
Entity
Text Type
Description
An instructor is a person who teaches a course.
If several persons teach a course, only the one person who is responsible
for the course and the grades counts as instructor.
Text Editor...
HTML Editor...
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (8)

"Text" Page:

- Textual descriptions/definitions of entities and attributes can be stored (in ASCII or HTML).
- Also "Notes" about entities and attributes can be stored, and the system can be extended to allow other text types.
- These texts will be part of the design documentation which can be generated by the "Repository Reports" Utility.

In Oracle, comments can also be stored in the data dictionary.


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Attributes
Name
Seq
Opt
Format
MaxLen
Dec
Primary
NAME
VARCHAR2
40
PHD_YEAR
NUMBER
4
0
PHONE
VARCHAR2
40
ROOM
VARCHAR2
128
STATUS
VARCHAR2
20
Insert Row
Delete Row
Reset Default
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (10)

"Attributes" Page:

- Here the entity attributes can be declared with the following information:
  - ◇ Name
  - ◇ Sequence number to define the order in which the attributes will be displayed (see below).
  - ◇ Domain, Data Type/Format (see below).
  - ◇ Is this attribute optional (i.e. possibly null)?
  - ◇ Is this attribute part of the primary key?
  - ◇ A short comment on the attribute.

Entity Properties (11)

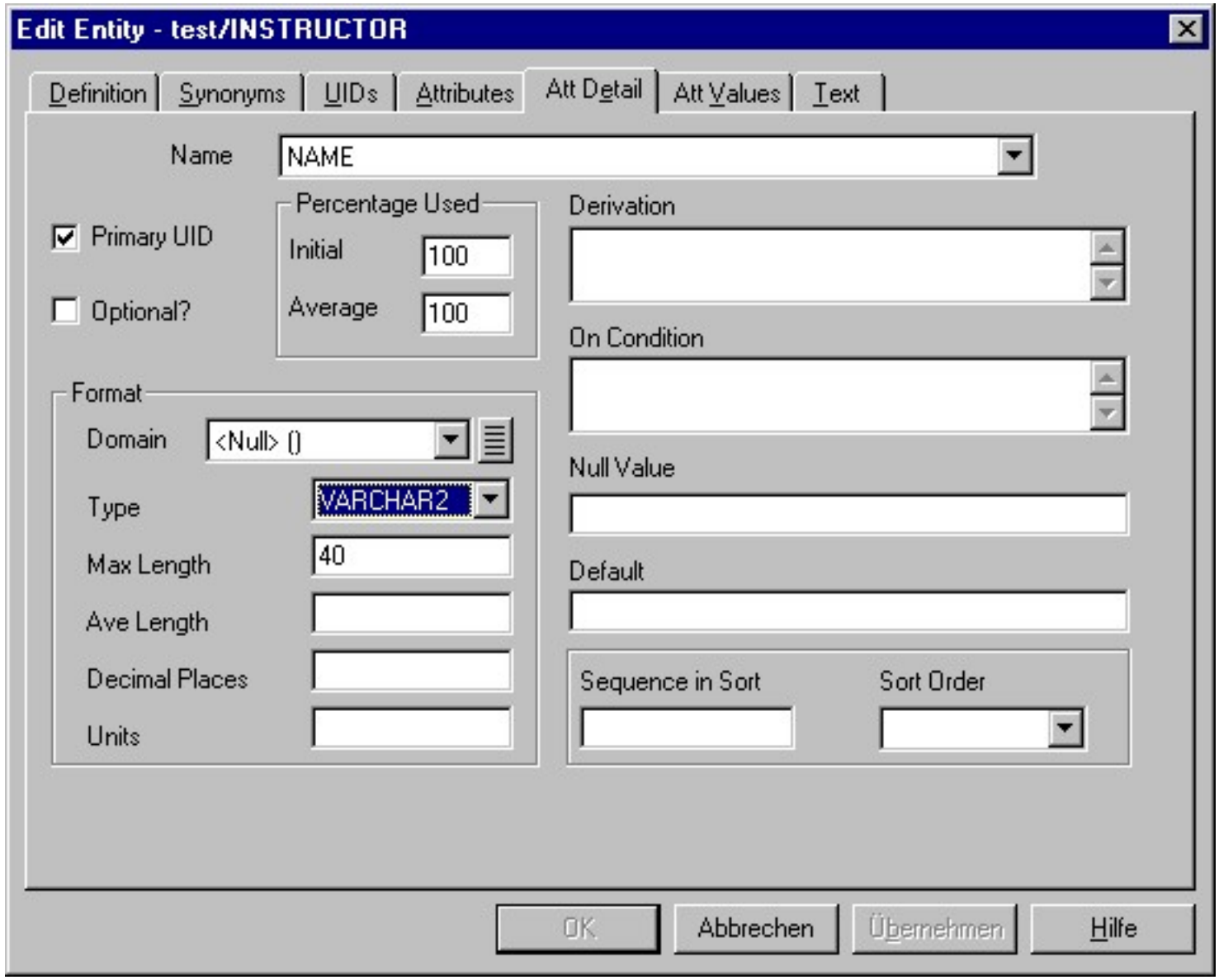
- Possible attribute formats (data types) are
  `CHAR`, `DATE`, `IMAGE`, `INTEGER`, `MONEY`, `NUMBER`, `PHOTOGRAPH`, `SOUND`, `TEXT`, `TIME`, `TIMESTAMP`, `VARCHAR2`, `VIDEO`.
- Not all of these data types exist in Oracle or all the other supported goal systems.
- It is a task of the logical design phase to map the data types used in the conceptual schema to data types supported in the selected DBMS.

  E.g. `IMAGE`, `VIDEO`, and `SOUND` would be mapped to a binary large object in Oracle (`BLOB`). The database design transformer contains mappings for various systems.

Entity Properties (12)

- Some types (e.g. `CHAR`, `VARCHAR2`, `NUMBER`) require a maximal length, some (e.g. `NUMBER`) also the number of decimal places after the point (precision, "dec").
- Instead of defining the data types for every attribute separately, one should use domains (see below).
- If the sequence number is left blank, one gets the default attribute sequence: (1) primary key attributes, (2) mandatory attributes, (3) optional attributes. Each group is alphabetically sorted.

  The alphabetical order is usually not what is intended.


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Name
NAME
Primary UID
Optional?
Percentage Used
Initial
100
Average
100
Derivation
On Condition
Format
Domain
<Null> ()
Type
VARCHAR2
Max Length
40
Ave Length
Decimal Places
Units
Null Value
Default
Sequence in Sort
Sort Order
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (14)

"Attribute Detail" Page:

- This tab has one page for each attribute.

  The attribute is selectable in the "Name" drop-down list.

- Each page contains all the information defined in the corresponding row of the "Attributes" table, plus more (see next two transparencies).

  Parameters shown on both pages are automatically updated on both pages when they are changed on one.

- No syntax checks are done on e.g. the derivation formula (any text can be entered, not only SQL).

  Also the default value is not checked against the type.

Entity Properties (15)

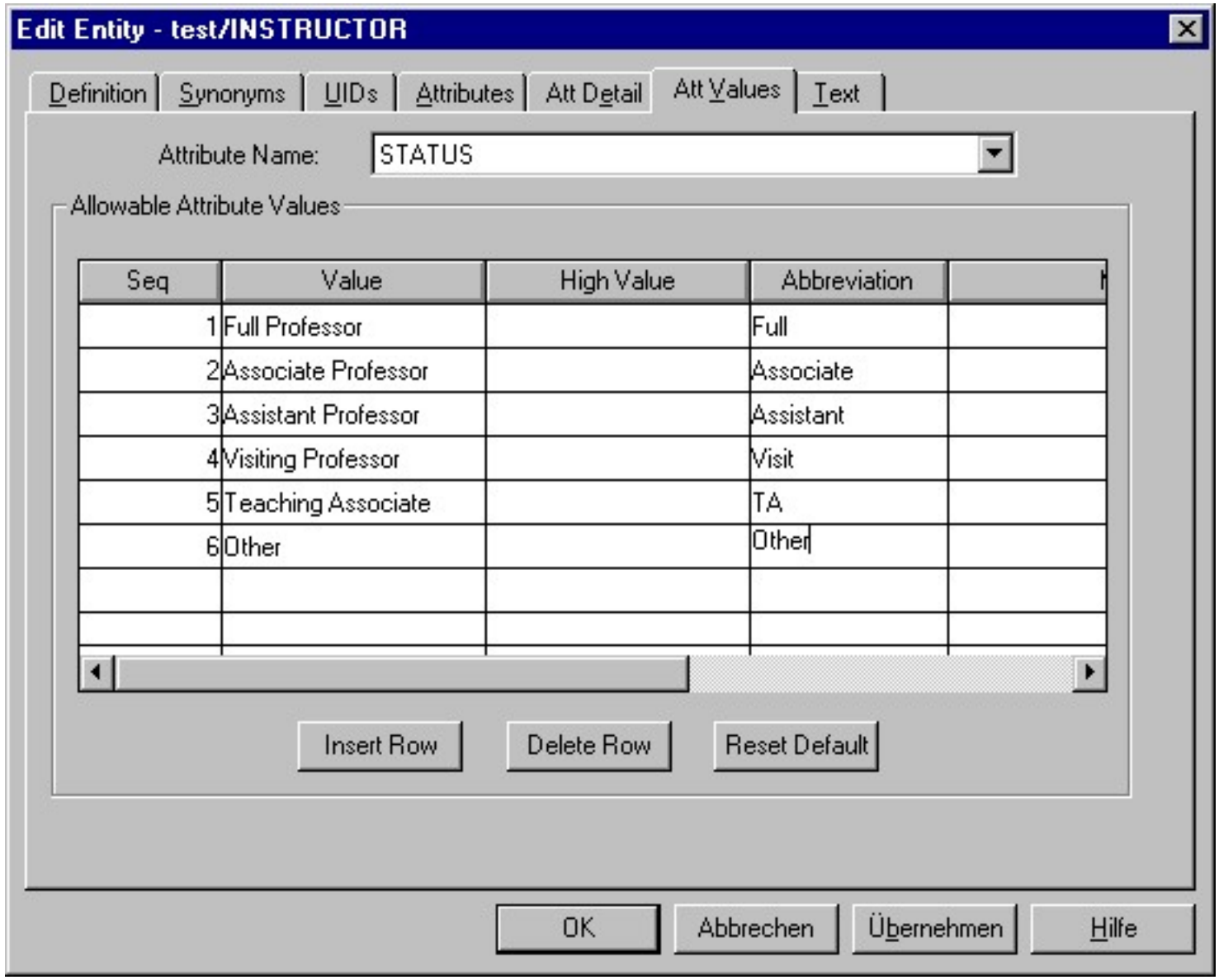
- Information on the "Attribute Detail" page:
  - ◇ Physical design information: average length and percentage of entities having a non-null value.
  - ◇ Units for the attribute (e.g. "kg", "cm", "in").
  - ◇ A derivation formula/algorithm if this attribute is derived.
  - ◇ A condition when this attribute can be used.

    E.g. "Flight Hours" is defined if/only if (?) "Job" is pilot. E.g. only associate and full professors can have a value in the column `TENURE_SINCE`.

Entity Properties (16)

- Information on the "Attribute Detail" page:
  - ◇ A representation for a null value if the DBMS does not support null values.

    This would be strange for a modern DBMS.
  - ◇ A default value (to simplify data entry).
  - ◇ If the entities/rows should be sorted by this attribute, the relative priority of this sort criterion and order (asc/desc).


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Attribute Name:
STATUS
Allowable Attribute Values
Seq
Value
High Value
Abbreviation
1 Full Professor Full
2 Associate Professor Associate
3 Assistant Professor Assistant
4 Visiting Professor Visit
5 Teaching Associate TA
6 Other Other
Insert Row
Delete Row
Reset Default
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (18)

"Attribute Values" Page:

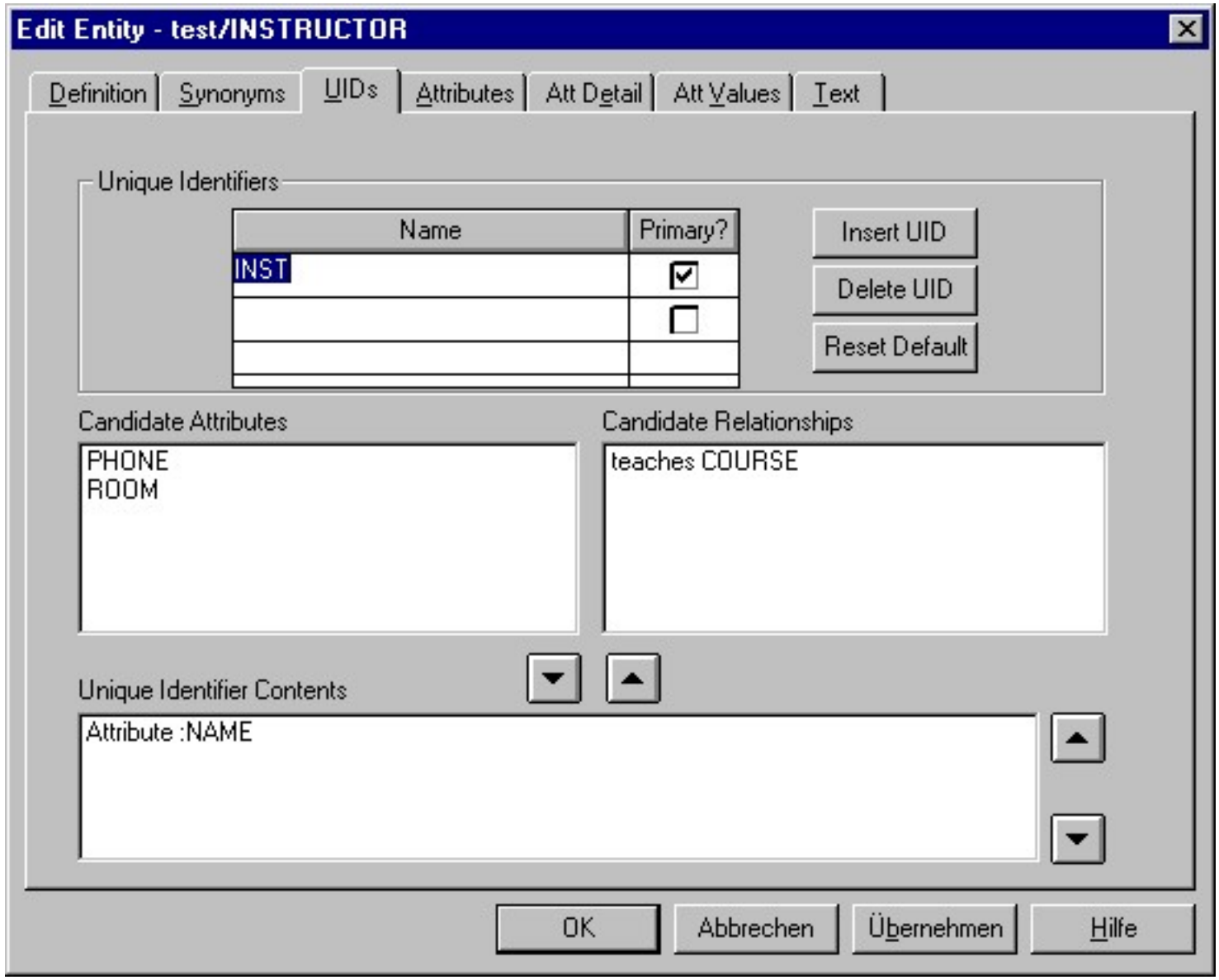
- On this page, restrictions for the values of an attribute can be defined (e.g. for "enumeration type" attributes).
- One can define all possible values of an attribute:
  - ◇ Value
  - ◇ Sequence number

    E.g. for printed documentation, menus in application programs.
  - ◇ Abbreviation
  - ◇ Meaning (help text)

Entity Properties (19)

- Already in the ER-design, information is collected that later can be used for the generation of application programs (forms for inserting data).
- Alternatively, one can define an interval of legal values.

  "Value" is the lower limit, "High Value" the upper limit. In general, the union of set of intervals is possible (by using several rows with "Value" and "High Value"), but this is hardly ever used.


Edit Entity - test/INSTRUCTOR
Definition
Synonyms
UIDs
Attributes
Att Detail
Att Values
Text
Unique Identifiers
Name
Primary?
INST
Insert UID
Delete UID
Reset Default
Candidate Attributes
PHONE
ROOM
Candidate Relationships
teaches COURSE
Unique Identifier Contents
Attribute :NAME
OK
Abbrechen
Übernehmen
Hilfe

Entity Properties (21)

"UIDs" Page:

- On this page keys (unique identifiers) can be defined.
- More than one key can be declared, but exactly one must be marked as primary key.

  Primary key information entered on this page is automatically reflected on the "Attributes" pages.

- The Designer does not prevent that a primary key attribute is optional (which is illegal in SQL).
- Each key/unique identifier must be named.

Entity Properties (22)

- Not only attributes, but also relationships can be used as a means for identification.

  Entity types that use this are also called weak entity types, see below.

- E.g. if instructors had a relationship to departments, and the UID consists of this relationship and the instructor name, there can be instructors with the same name in different departments.

  The foreign key that contains the ID of the department together with the instructor name becomes a composed key of the instructor. This works only for relationships with a $(1,1)$-cardinality, e.g. on the many side of a one-to-many relationship. The Designer does not check this.

Domains (1)

- Often different attributes should have the same data type, i.e. especially the same length. E.g.:
  - ◇ Years: Year an instructor got tenure, Year a course is offered, Year a student was admitted, etc.
  - ◇ URLs: Links to homepages of courses, instructors, departments.
  - ◇ Last Names: Of students, instructors, staff.
- It would be strange if some years are stored with two digits, others with four, or student names can be longer than instructor names.

Domains (2)

- Characteristics such as the maximal length of all kinds of URLs should be defined only once.
- This ensures greater consistency in the schema, especially when later changes are done (e.g. attribute length increases).
- In Oracle Designer, one can define data types of columns indirectly via domains:

Column "Homepage" → Domain "URL" → Data Type "VARCHAR(80)"

Domains (3)

- One first defines a domain and then assigns this domain to one or more attributes.

  Instead of directly defining the data type details for the attributes. That would have to be done for each attribute separately, while with the domain the details are defined only once and used in possibly many attributes. In Oracle Designer, domains are defined under "`Edit` → `Domains`".

- If a domain definition changes, one can propagate this change to all attributes having this domain.

  In Oracle Designer, this is done only semi-automatically. One must call "`Utilities` → `Update Attributes in Domain`".

Domains (4)

- Different domains may have same data type.
- E.g. last names of customers and names of cities may both be `VARCHAR(20)`, but it makes no sense to compare them. Different domains should be used.

  One should consider attributes of different domains as uncomparable (unless declared as subtype).

- A domain can be seen as a "shorthand" for a standard data type, but with a specific meaning, different from other domains.

Domains (5)

- Domains can be used to capture the information which attributes should be comparable.

  This requires logical domain names, e.g. `CITY`, not `VC20`.

- The SQL-92 standard has a similar notion of domains (without the restriction that columns of different domains cannot be compared).

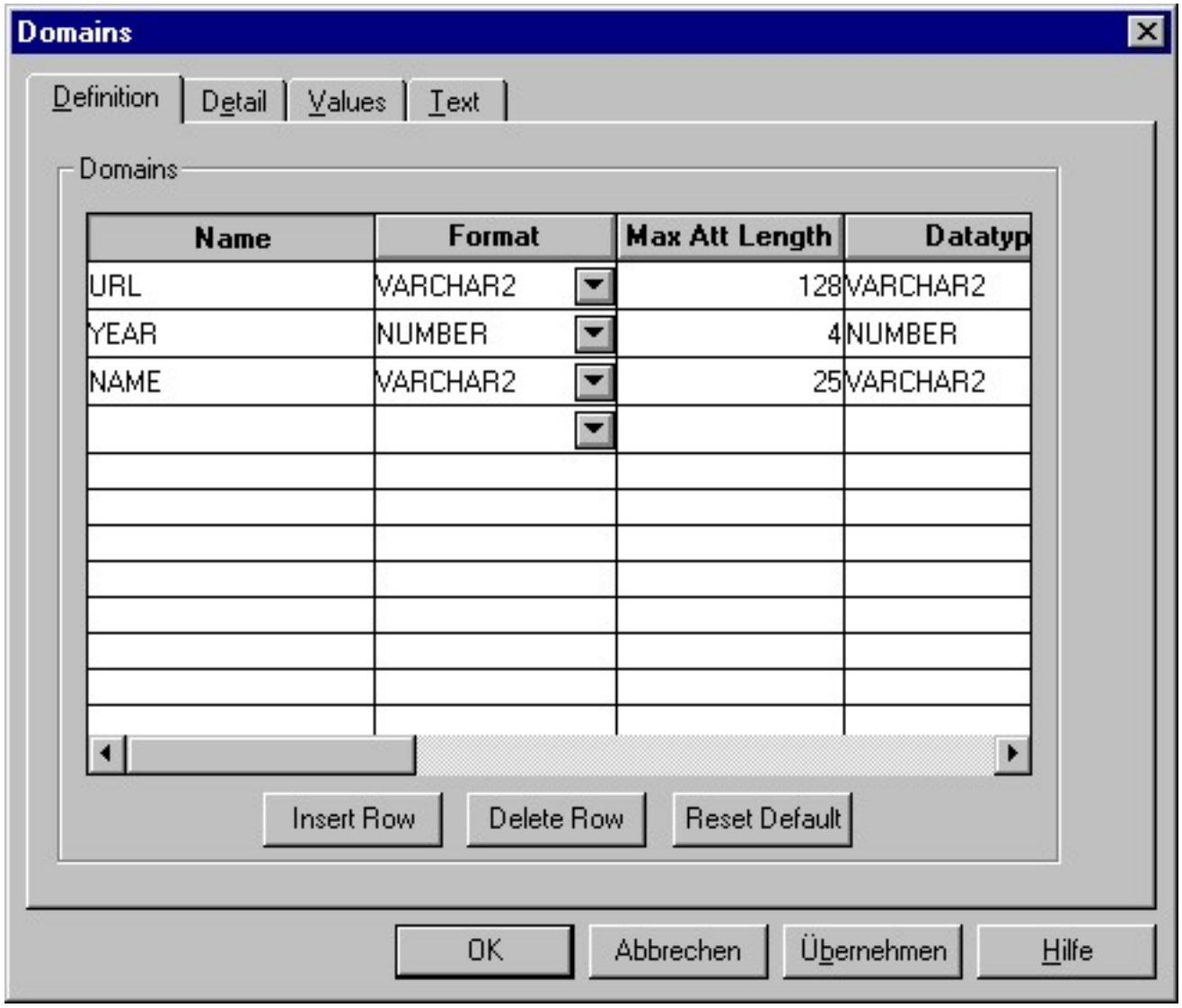
  This is not implemented in Oracle 8. But when domains are defined in the Designer, they might be partially mapped to SQL domains in other DBMS. Oracle 8 has PL/SQL types which could also be used. But for consistent schema changes, it is already helpful that they are supported in the Designer.

Domains (6)

- Domain names can often be used as attribute names. This makes joinable attributes very clear.
- Some designers have a set of standard domains, which they always use.

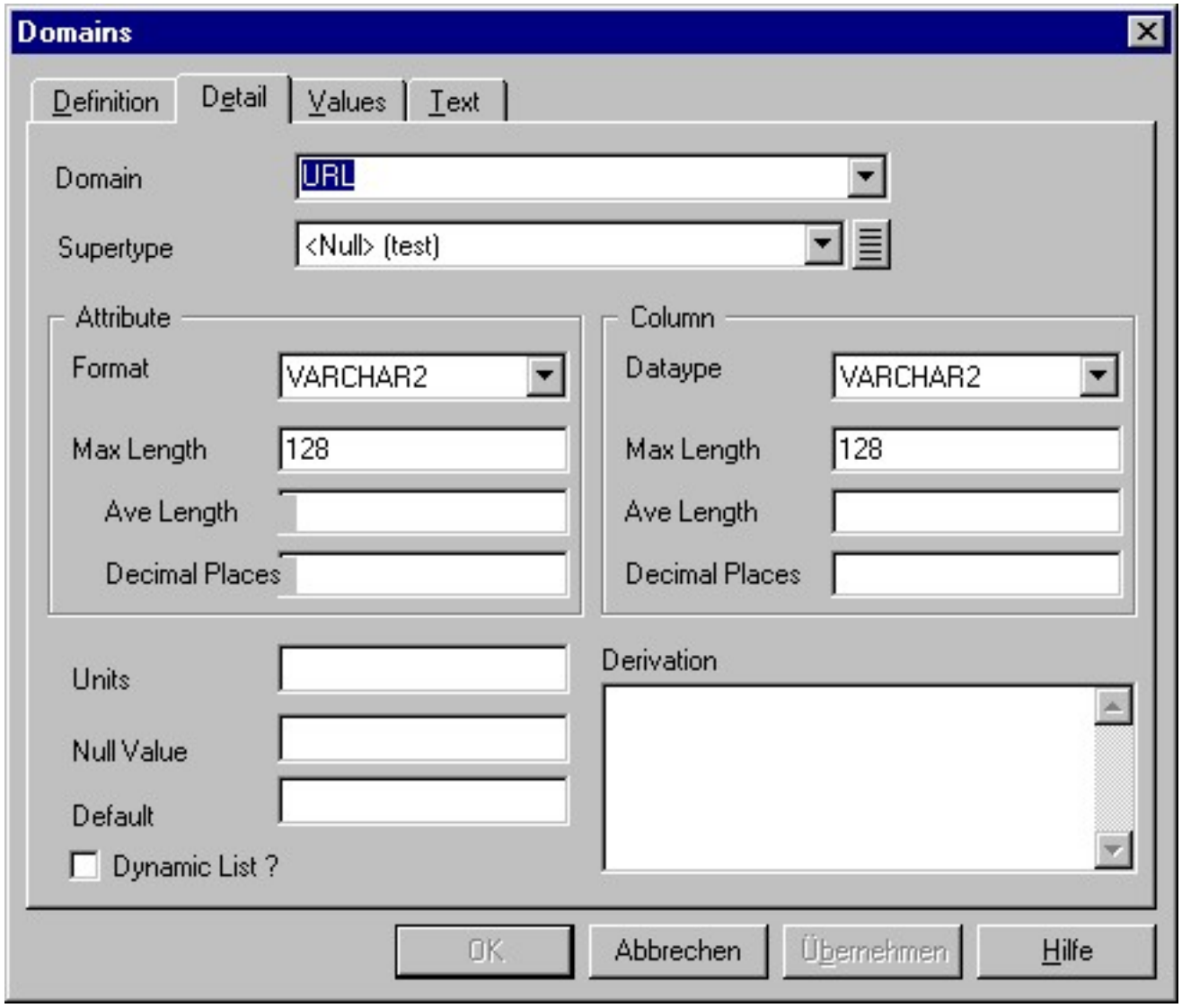
  E.g. names of length 10, 20, 40, descriptions of size 2000, email/URL of size 250, ZIP codes, SSN, boolean values, etc. Selecting from a set of predefined standard domains can be done faster than considering every attribute in isolation. In some projects, only a "domain administrator" is allowed to create a new domain.

- However, this at least partially contradicts the idea of logical domains.


Domains
Definition
Detail
Values
Text
Domains
Name
Format
Max Att Length
Datatyp
URL
VARCHAR2
128
VARCHAR2
YEAR
NUMBER
4
NUMBER
NAME
VARCHAR2
25
VARCHAR2
Insert Row
Delete Row
Reset Default
OK
Abbrechen
Übernehmen
Hilfe

Domains (8)

- The dialog box for defining domains has four tabs:
  - ◇ “Definition”: list of all defined domains.
  - ◇ “Detail”: one page per domain.
  - ◇ “Values”: for defining enumerated types etc.
  - ◇ “Text”: contains descriptions, notes, etc.
- In principle, the same parameters can be set as in the attribute definitions.
- Whether an attribute is optional and whether it is part of the key can only be defined in the entity definition dialog.


Domains
Definition
Detail
Values
Text
Domain
URL
Supertype
<Null> (test)
Attribute
Format
VARCHAR2
Max Length
128
Ave Length
Decimal Places
Column
Datatype
VARCHAR2
Max Length
128
Ave Length
Decimal Places
Units
Null Value
Default
Dynamic List ?
Derivation
OK
Abbrechen
Übernehmen
Hilfe

Domains (10)

Format/Attribute vs. Datatype/Column:

- A domain definition contains information at the ER-level (Format) as well as about the implementation (Datatype).

  The documentation also mentions that the datatype can also be used for application program variables, but then it depends on the programming language. The type system of languages like C are quite different from the SQL type system.

- E.g. "IMAGE" can be selected on the conceptual level, but it is implemented as a "BLOB".

  BLOB: "Binary Large Object". The Datatype selector contains all datatypes of Oracle as well as some types from other systems.

Domains (11)

Dynamic List:

- If this is checked, the possible values will be retrieved at runtime from a lookup table.

  This makes it easy to change the possible values of the enumeration type later: One can simply insert a new value into the lookup table.

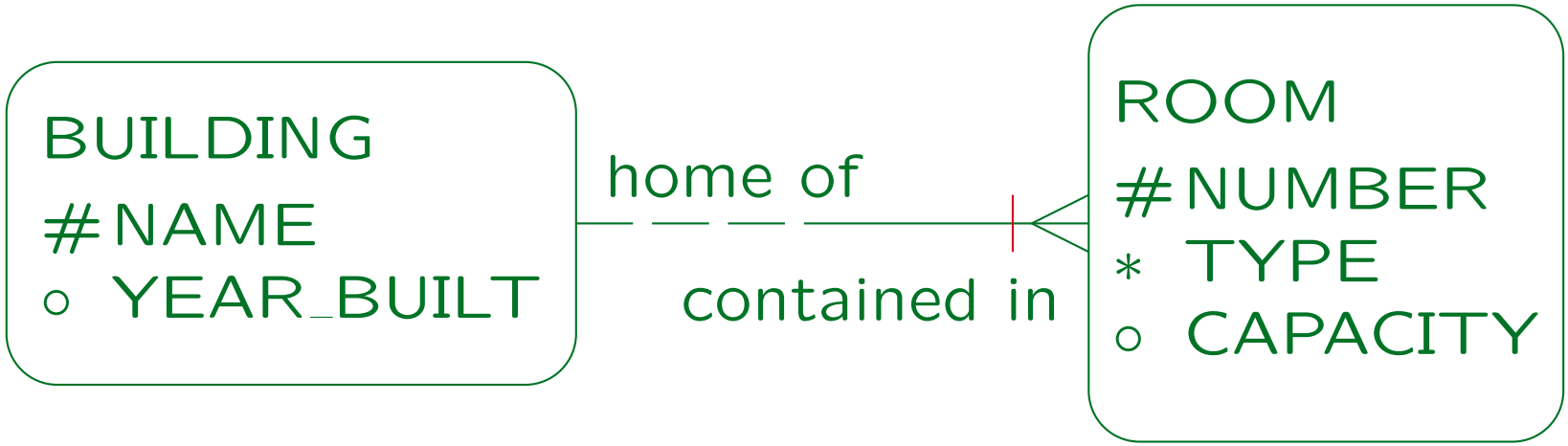
- Otherwise, they will be hardcoded (e.g. as `CHECK`-constraint in the `CREATE TABLE` statement).

  While an `ALTER TABLE` statement to change the constraint is not too difficult (but not that several attributes in different tables might have to be changed), the possible values might also be hardcoded in application programs.

Overview

Weak Entity Types (1)

- If a relationship contributes to the identification, there is a bar across the connecting line:



- A room is identified by building and room number, e.g. “Crawford Hall 169”.

Weak Entity Types (2)

- Different buildings of the university can have rooms with the same number.
- When translated into tables, the key of the ROOMS table will be composed from the building name and the room number.

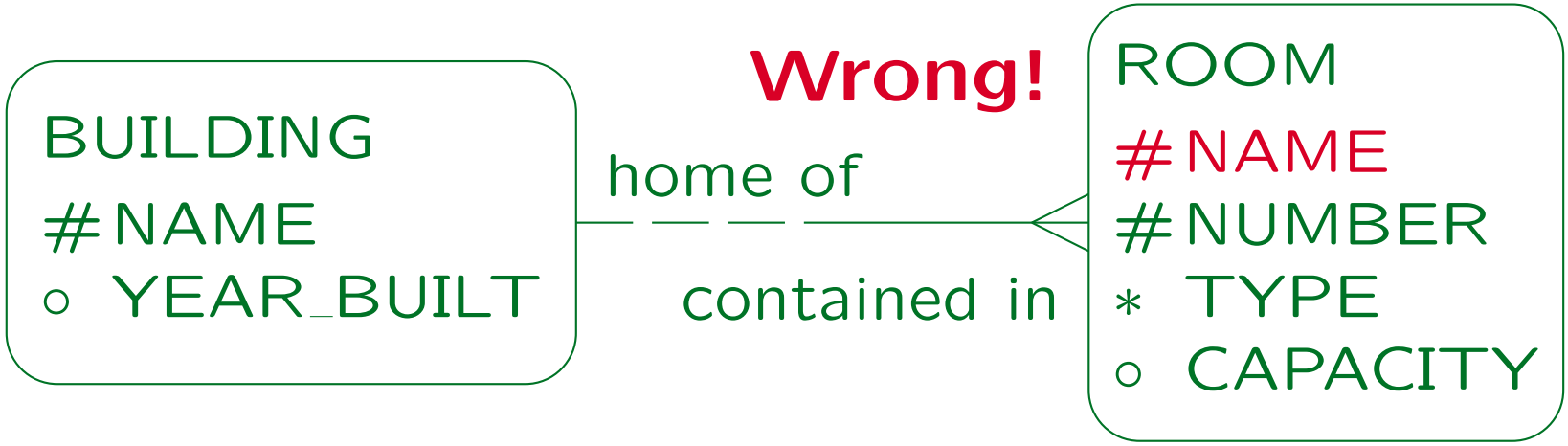
  The building name in addition will be a foreign key that references the BUILDINGS table.

- Entity types that must borrow key attributes from other entity types are called "weak entity types".

  In the Oracle Designer documentation, this name is not used. One simply declares a relationship as part of a UID for an entity type.

Weak Entity Types (3)

- It would be a bad design to explicitly replicate the key attribute of the referenced table:



- Now the constraint is needed that a room with name X is always related to a building with name X, so that the relationship is actually redundant.

Weak Entity Types (4)

- In general, advanced constructs in the ER-model are often introduced in order to avoid certain common kinds of constraints.

  Or at least to specify these constraints graphically instead of as text and permit a special implementation. If one would translate the above schema where name and number are explicitly defined as key attributes into the relational model, one would get two copies of "Name": A second copy is introduced as foreign key in order to implement the relationship (see below). Now with the constraint it becomes clear that the two copies can be merged.

- Weak entity types are often used in master-detail relationships, e.g. for an invoice and its line items.

Weak Entity Types (5)

- A weak entity type is existence-dependent on its parent entity type (BUILDING in this case): If a building is sold and removed from the database, all rooms in it should be automatically removed.

  For weak entity types, it is quite typical that in the resulting relational schema "`DELETE CASCADES`" is defined for the foreign key that implements the relationship.

- It is often a design decision how one selects a key: If rooms have a number that it unique over all buildings, the "Room" entity type is no longer weak.

Weak Entity Types (6)

- One can use a relationship for identification only if there is at most one related entity (cardinality $(1,1)$ or $(0,1)$):
  - ◇ On the many side of a one-to-many relationship.
  - ◇ On both sides of a one-to-one relationship.

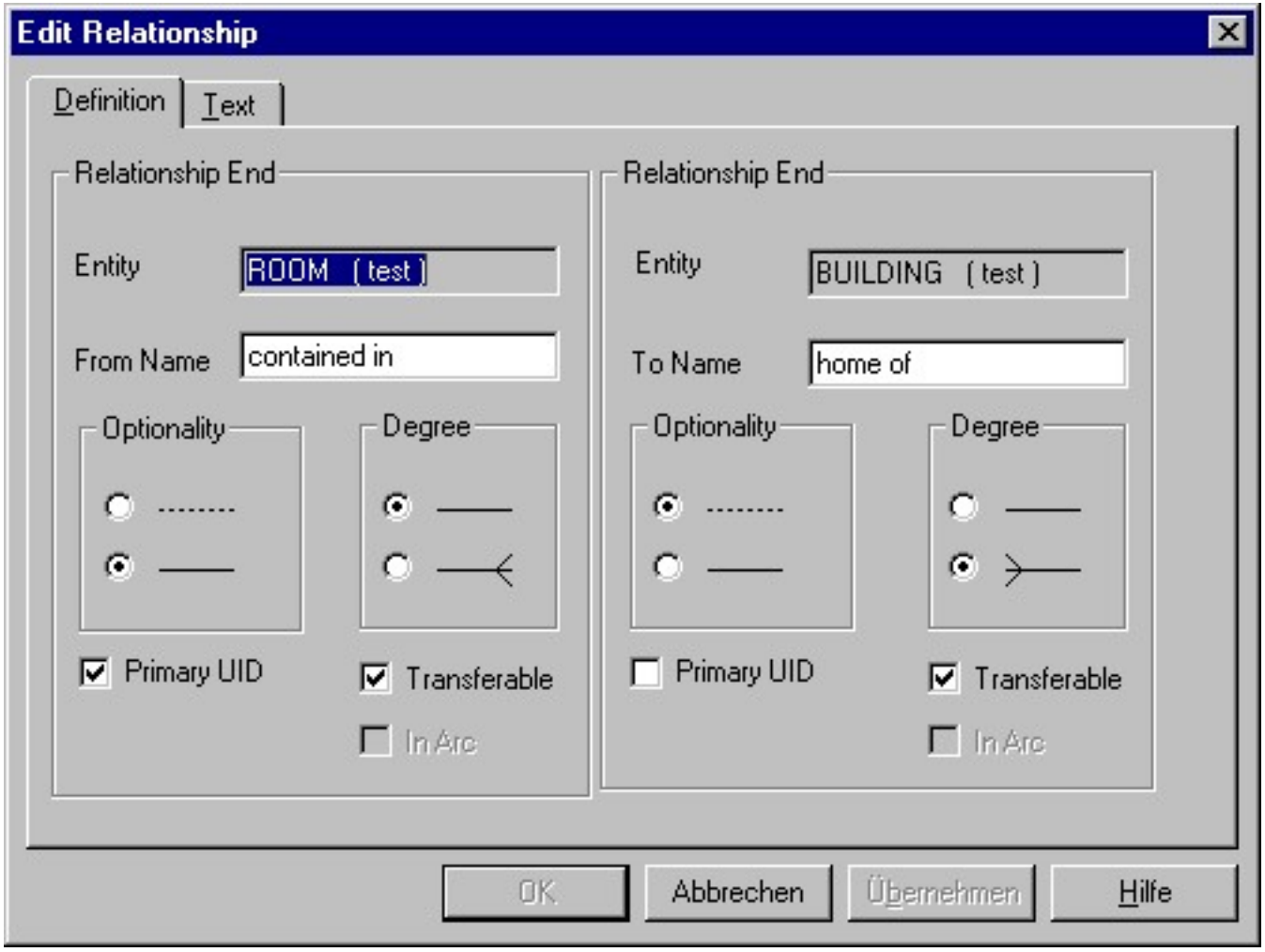
  If there were several related entities, one would need set-valued attributes (not supported in the standard relational model).
- At least for primary keys, the participation in the relationship must be mandatory.

  Primary key attributes cannot be null.

Weak Entity Types (7)

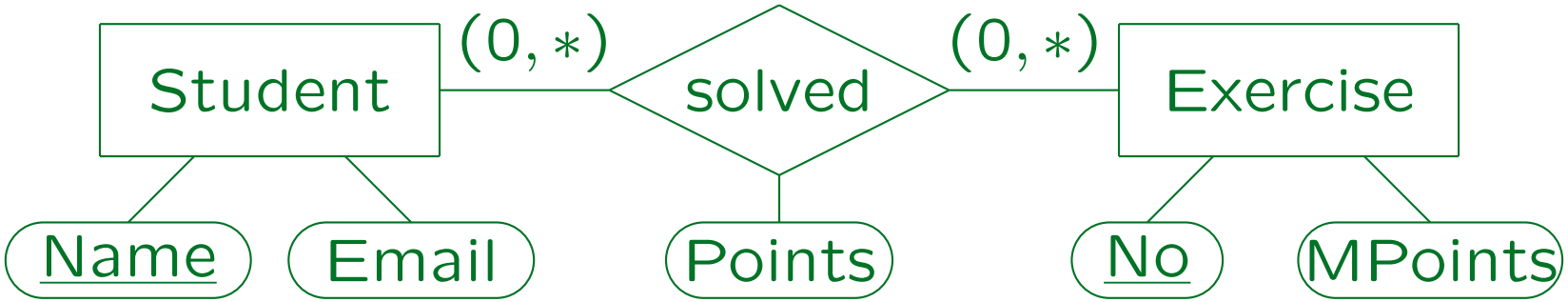
- There are two places to specify that a relationship contributes to the identification of the entity:
  - ◇ In the entity definition, tab UIDs.
  - ◇ In the relationship definition.

    In the "Edit Relationship" dialog box, one can also change the optionality (minimum cardinality) and degree (maximum cardinality) for each end, change the role name, and store a description or notes for each relationship end.


Edit Relationship
Definition
Text
Relationship End
Entity
ROOM (test)
From Name
contained in
Optionality
Degree
Primary UID
Transferable
In Arc
Relationship End
Entity
BUILDING (test)
To Name
home of
Optionality
Degree
Primary UID
Transferable
In Arc
OK
Abbrechen
Übernehmen
Hilfe

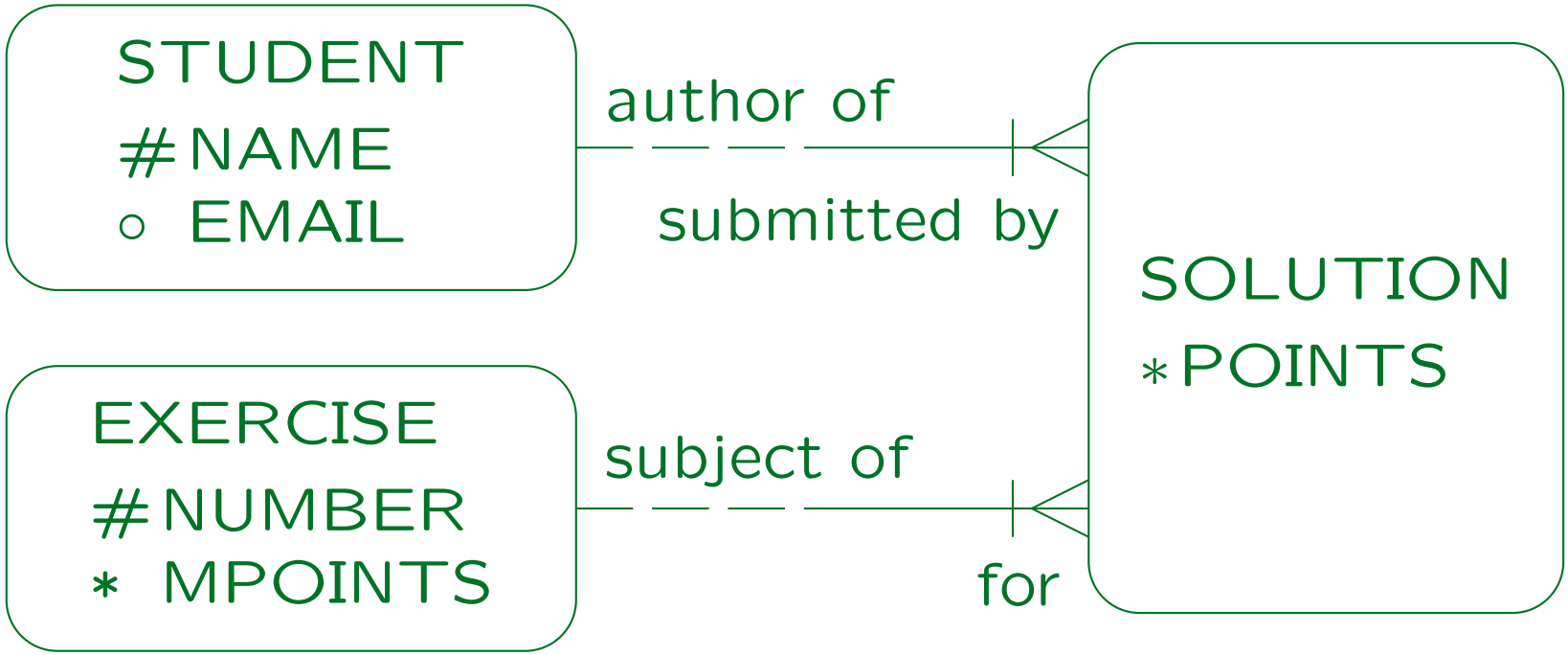
Association Entity Types (1)

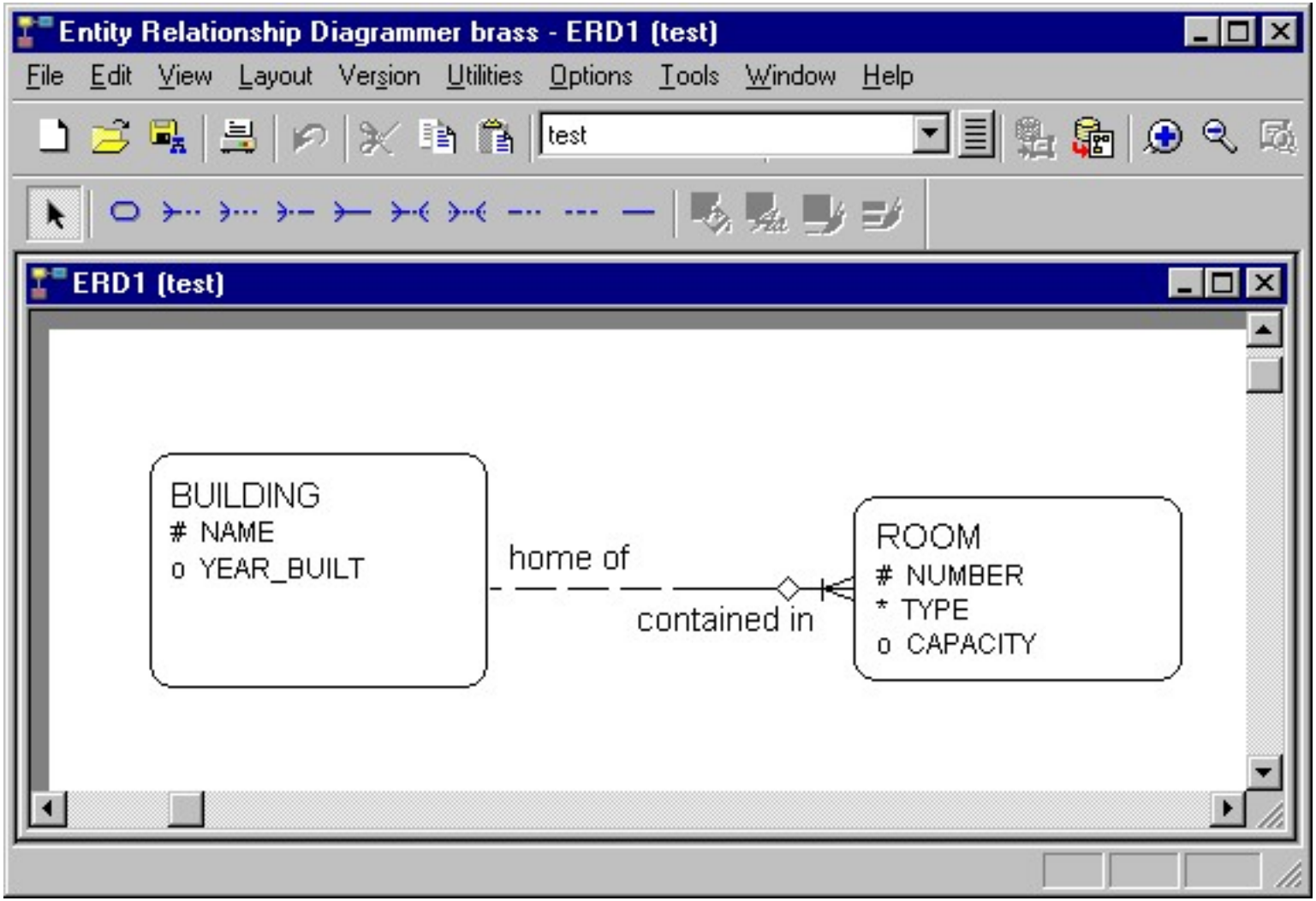
- Weak entity types can have more than one parent.
- Weak entity types with two (or more) parent types are sometimes called "association entity types", because they are similar to a kind of relationship between the parent entity types.
- E.g. suppose that we need a relationship attribute:



Association Entity Types (2)

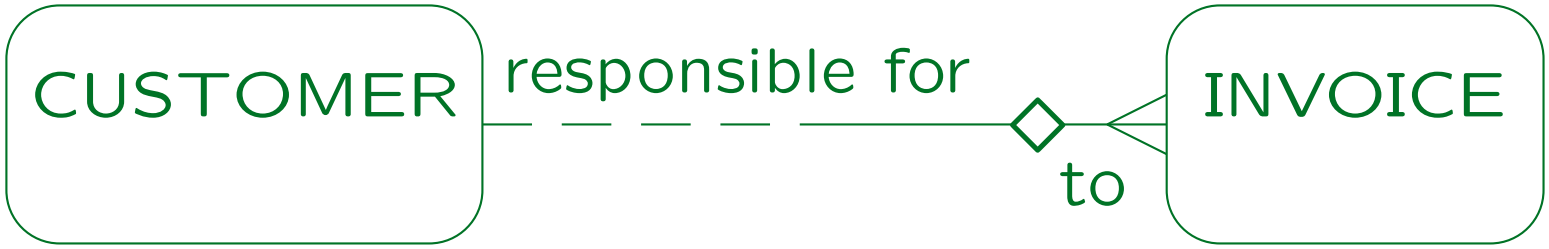
- Oracle Designer does not support relationship attributes. However, one can turn the relationship into an association entity type:




Entity Relationship Diagrammer brass - ERD1 (test)
File Edit View Layout Version Utilities Options Tools Window Help
test
ERD1 (test)
BUILDING
NAME
o YEAR_BUILT
home of
contained in
ROOM
NUMBER
* TYPE
o CAPACITY

Fixed Relationships (1)

- A relationship can be marked as non-transferable:



- In this way, an invoice cannot be disconnected from a customer and connected to another customer.
- I.e. the foreign key attribute (customer number in the invoice) is non-updatable.

  Oracle Designer allows the "non-transferable" sign also on the other side of the relationship. Semantics unclear (?).

Fixed Relationships (2)

- It might be a good idea to collect non-updatability information for arbitrary attributes, but Oracle Designer does not allow that.

  However, one can extend it in this way. It also has cross-referencing tools which show CRUD (create, retrieve, update, delete) information for all entities based on the business functions.

- Non-Updatability is a simple kind of dynamic constraint, which refers not to single database states as a normal static constraint, but to pairs of DB states.

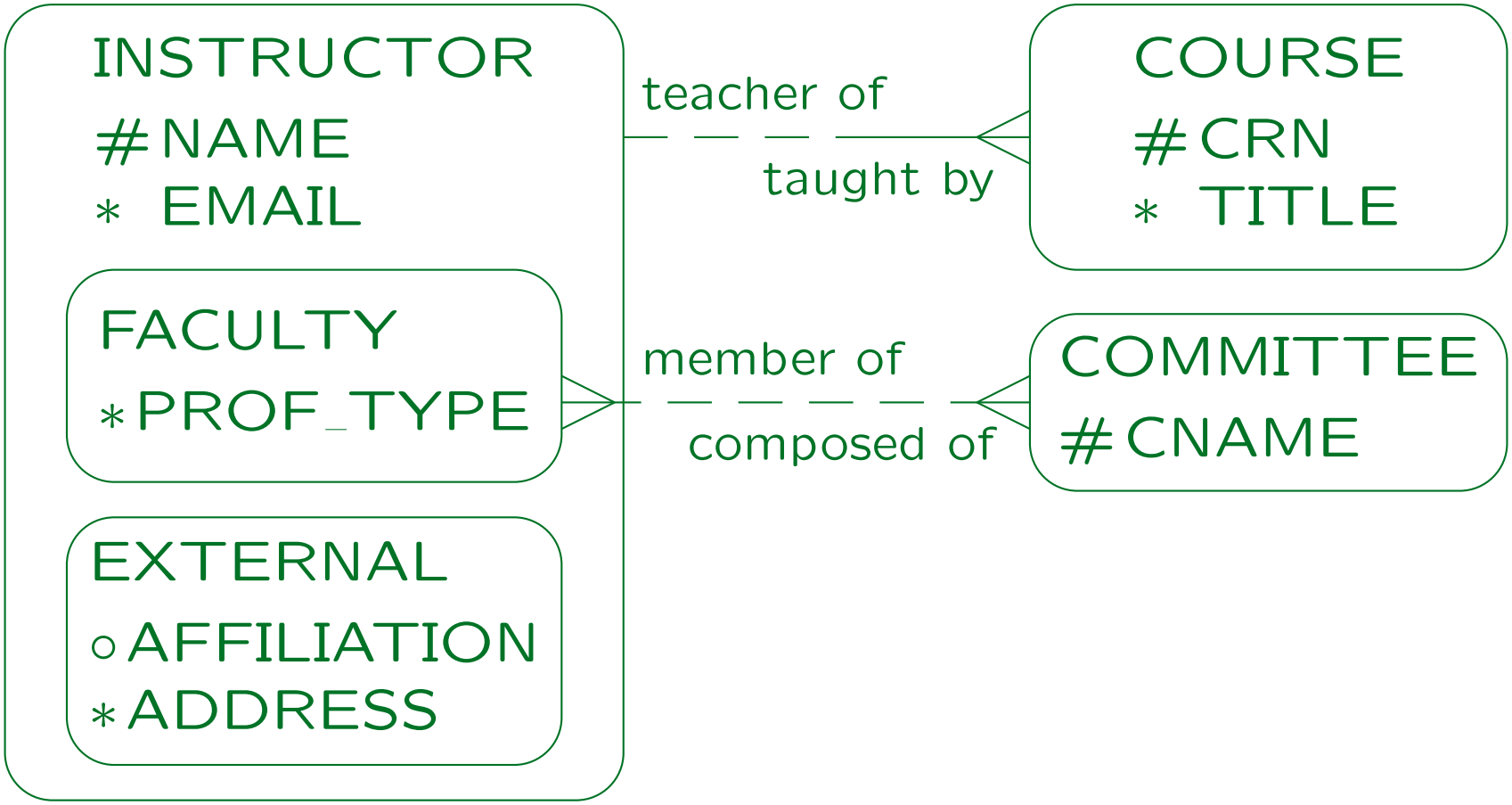
  Another example is "Salaries cannot decrease."

Overview

Specialization (1)

- Two or more entity types may have attributes or relationships in common.
- Then it might be useful to create a generalized entity type, which contains only the common characteristics, and abstracts from the differences.
- Or, some attributes or relationships may apply to only a subset of the entities. Then creating a specialized entity type for this set should be considered.
- Inheritance ("is-a" relationships) and subclasses are also a useful feature of object-oriented languages.

Specialization (2)



Specialization (3)

- In the above example:
  - ◇ Instructors are faculty members (i.e. long-term employees of the university) or external teachers which are paid for the course only.
  - ◇ For all instructors, name and email address is stored.
  - ◇ For faculty members, in addition the professor type (Assistant, Associate, Full) is stored.
  - ◇ For external instructors, their affiliation and address is stored.

Specialization (4)

- Example, continued:
  - ◇ Both types of instructors can teach courses.
  - ◇ Only faculty members can serve on committees.
- In general, specialization can be distinguished in:
  - ◇ Disjoint: It is not possible that an instructor is at the same time a faculty member and an external teacher. Oracle Designer only supports this case.
  - ◇ Overlapping: Objects of the superclass can be in more than one subclass at the same time (then they do not have a unique type: uncommon).

Specialization (5)

- Specialization can also be:
  - ◇ Total: Every instructor must be a faculty member or an external teacher.
  - ◇ Partial: Elements of the superclass are not necessarily contained in one of the subclasses.
- Oracle Designer normally uses total specialization (but one can always create an "Other" subclass).
- However, when one generates tables, one can also select that the supertype is instantiable (meaning partial specialization).

Specialization (6)

- Total and disjoint specialization means that the set of entities of the superclass is partioned into the instances of the subclasses.

  It is very difficult to find information like "Oracle Designer supports only non-overlapping and total specialization" in the documentation. E.g. it is not mentioned in the online help, the manuals are anyway either too short or only interface lists, and books like the Oracle Designer Handbook assume that you know ER-modelling. Only the book by Barker clearly states this. Looking at the translation into tables also shows that a non-overlapping and total specialization is assumed. I later learnt about the option for the Database Design Transformer which gives partial specialization, but this is anyway the wrong place: If such an option is really to be used, it must be offered in the ER-Diagrammer.

Specialization (7)

- Subclasses can themselves have subclasses.

  In the ER-Diagrammer, one creates a subclass by creating an entity type within the boundaries of another entity type (the superclass).

- In general, one can create a tree of entity types.

  When specialization is total, real instances only exist at the leaves of the tree. All other classes in the tree simply have the union of the members of their subclasses.

- Multiple inheritance is not supported in Oracle Designer.

  Multiple inheritance means that an entity type has more than one superclass, and inherits attributes and relationships from all of them.

Specialization (8)

- It makes no sense to define primary key attributes for a subclass: All attributes and the key constraint are inherited from the superclass.

  If the key uniquely identifies all members of the superclass, it especially uniquely identifies the members of the subclass.

- Of course, it is possible to declare additional (secondary) keys for the subclasses.

Specialization and Null Values

- In principle, one could avoid optional attributes with specialization: E.g. "Instructor" has a subclass "Instructor with Home Phone Number".
- When there are n attributes that can independently be null, one would need 2^n subclasses. Then this method is clearly not useful.
- However, when there is a group of attributes which can only be together null, or together not null, one should consider using a subclass.

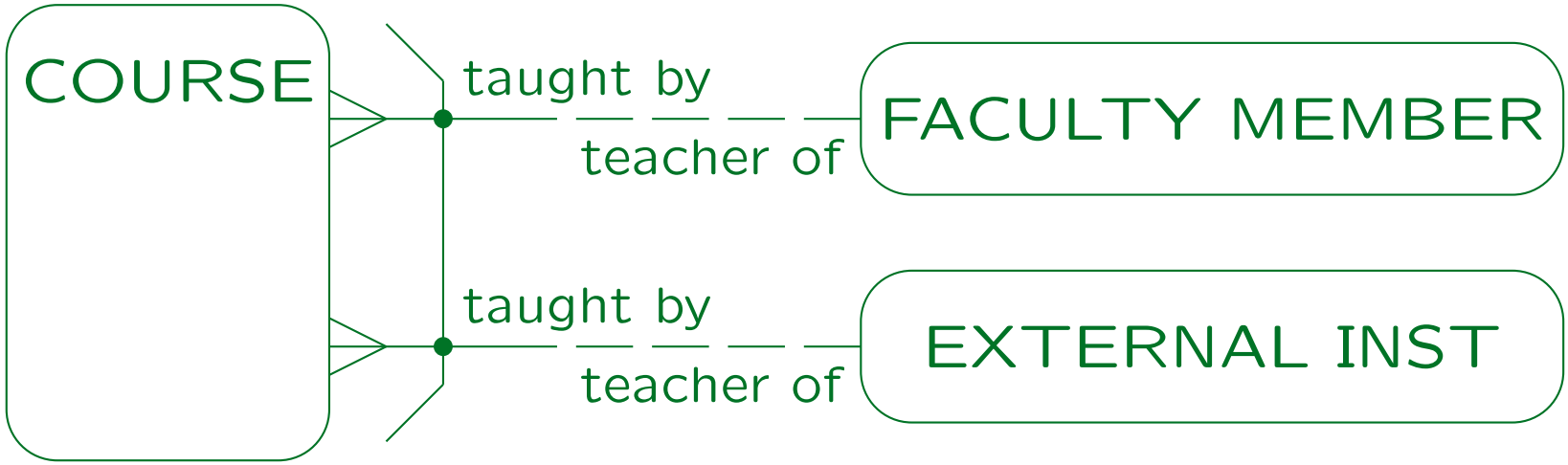
Constructs like specialization reduce the need for constraints.

Generalization

- The specialization process starts with the superclass, discovers that some attributes apply only to a subset of the entities, and constructs subclasses.
- Vice versa, in generalization the subclasses are identified first, and then the discovery of common attributes leads to a superclass. The result is identical.
- Some authors use the term generalization or categorization if the subclasses have keys of their own, and their union should be considered, e.g. for defining a relationship.

Arcs (1)

- By linking two or more relationships with an arc, one can specify that they are mutually exclusive:



- I.e. a course is either taught by a faculty member or by an external instructor, but not by both.

Arcs (2)

- This is similar to defining two subclasses of courses:
  - ◇ Courses that are taught by a faculty member.
  - ◇ Courses taught by an external instructor.
- Alternatively, this corresponds to a generalization of faculty members and external instructors.

  One would use this model e.g. if external instructors and faculty members already have different keys of their own, and there is no natural key for their generalization. This is not a good example: The name or SSN would do. The classical example in the literature are invoices which can be sent to persons or companies.

Arcs (3)

- In general, arcs might help when for various reasons specialization is too restricted.
- Using arcs in the ER-Diagrammer is a bit tricky.

  Arcs are created by selecting at least two relationship ends and then clicking on the "create arc" symbol in the toolbar (or the Utilities menue). You must select the relationship ends, not the relationships (click on the role names). Use `Ctrl`-click to select the second end.

  In order to remove a relationship from an arc, select the arc by clicking on the line between the two relationships (this is a bit difficult). Then select the relationship end(s) you want to remove and select "Remove from Arc" on the toolbar or the Utilities menu. If an arc remains only for one relationship, I do not know how to select it. In this case, use the Repository Object Navigator, drill down to the relationship, and delete the "1" in the field "In Arc".