Part 2: ER-Diagrams in Oracle Designer

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.
- Oracle Designer Model, Release 2.1.2 (Element Type List).
- Oracle Designer Online Help System.
- 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.
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 2, Oldenbourg, 1997.

Objectives

After completing this chapter, you should be able to:

- write a short paragraph about Oracle Designer.

  What is it, what are its main components, and how does it support the design and development of database application systems?

- draw ER-diagrams in the graphical syntax of Oracle Designer.

  With and without actually using the tool. You should also be able to read such diagrams, and to enumerate the supported ER-constructs.

- explain the difference between the global DB schema and the views contained in single diagrams.
- explain the role of the repository.

Overview

Oracle Designer (1)

- Oracle Designer is a CASE tool.

  CASE = Computer Aided Software Engineering. It was earlier called Oracle CASE*Designer, then renamed to Designer/2000 and is now called Oracle Designer.

- Some people also call it "documentation tool".

  The main goal of database design tools is to collect all design information (documents) in one place.

- Oracle Designer is a large software package that helps to design databases and application programs (together: DB Application Systems).

  It is not a database system.

Oracle Designer (2)

- E.g., it contains a graphical editor for ER-diagrams, called the "Entity-Relationship Diagrammer".

  But that is not more than 10% of the entire tool.

- Not all design information that Oracle Designer manages is visible in the diagrams.

  E.g. when one clicks on an entity type, a dialog box opens that permits to store further information.

- Oracle Designer can also be used to design databases for non-Oracle systems.

  Oracle's application programming tools are specially supported, but the pure DB design works well for other systems.

Oracle Designer (3)

- Oracle Designer supports the following types of diagrams for modelling system requirements:
  - Business Process Diagrams
  - Entity-Relationship Diagrams
  - Dataflow Diagrams
  - Function Hierarchy Diagrams
- In addition, there are two types of design diagrams:
  - Server Model Diagram (Relational DB Schema)
  - Module Diagram (Application Programs)

Oracle Designer (4)

- Oracle Designer has also tools that generate
  - relational schemas from ER-schemas (Database Design Transformer),

    It can also produce DDL code (schema definitions) for DBMS from other vendors, not only Oracle.

  - first-cut application programs (Application Design Transformer).

    Code can be generated for stored procedures/triggers (PL/SQL), Oracle Developer (Forms), Oracle Report Builder (produces reports in PDF and HTML format), Oracle Application Server (web interface consisting of PL/SQL Procedures), C++, Visual Basic, MS Help.

Oracle Designer (5)

- Oracle Designer has also tools for capturing and reverse engineering the design of an existing DB.

  This takes the information from (a) the data dictionary of an Oracle DB, or (b) from DDL files, or (c) via ODBC calls.

- Also the design of certain pre-existing application programs can be extracted and added to the Designer information.
- Oracle Designer has also tools for creating reports about the design (design documents) out of the collected information, cross-referencing tools, etc.

Oracle Designer (6)

- All Designer tools are integrated: E.g. entities created with the ER-Diagrammer can be referenced in dataflow diagrams.
- Oracle Designer supports work in all phases of the system development lifecycle.

  Tools supporting only the first phases are called "Upper CASE Tools". In contrast, physical design information, storage information, database users etc. can be specified in Oracle Designer so that the database can be completely generated out of the collected information. Also a large part of application program development can already be done inside Oracle Designer.

Oracle Designer (7)

- Oracle Designer supports groups of designers developing a database application system together.

  This is based on the multi-user features of the underlying database system. If the "Broadcast Server" is installed, users are automatically notified when objects in their diagrams are changed by other users.

- It also has support for version management.

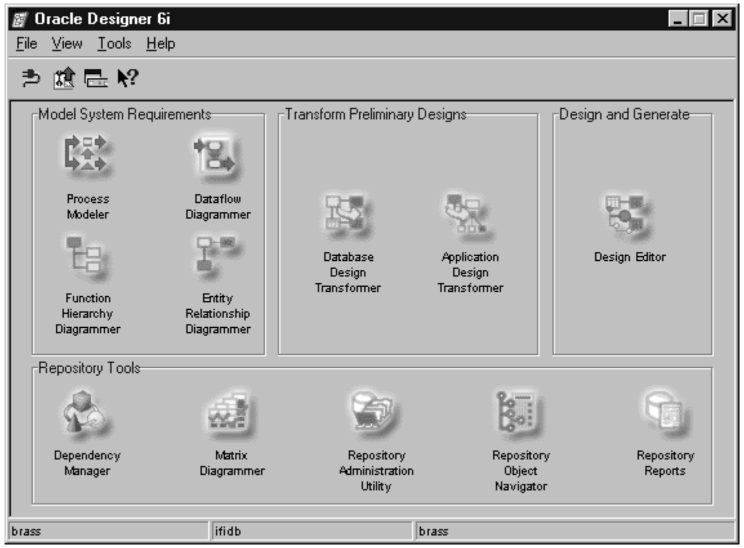
  This was significantly extended in version 6i.

- UML is not supported in Oracle Designer.

  There was an extension of Oracle Designer called "Object Database Designer" (ODD) which supported a subset of UML. It was shipped e.g. with Designer 2.1.2, but it is not being further developed and not included in Designer 6i. Oracle JDeveloper supports UML.

Oracle Designer (8)

- Oracle Designer runs only on Windows.
- From the Tutorial:
  - ◇ "Oracle Designer is a software tool for analyzing business requirements and for designing and generating client/server systems that meet those requirements. Oracle Designer incorporates support for business process modeling, systems analysis, software design and system generation."
  - ◇ "Oracle Designer provides a multi-user repository and is closely integrated with Oracle Developer, Oracle's client / server development toolset. In this way, Oracle Designer allows organizations to design and rapidly deliver scalable, client/server systems that can adapt to changing business needs."



Repository (1)

- Oracle Designer stores the design information not in files, but in an Oracle database.

  Also the diagrams are stored in the database. Whereas one can design also non-Oracle databases, the design information must be kept in an Oracle database.

- The set of all tables etc. used for storing this information is called the repository.

  The Designer 2.1.2 repository contains 1532 objects, of which are 107 tables, 273 views, 149 indexes, 64 constraints, 17 triggers, 9 sequences, 472 packages, 510 package bodies.
  In Designer 6i, the repository consists of 5549 (6i Release 1) or 5561 (Release 2 and 3) objects. Release 1: 392 tables, 757 indexes, 720 views, 1008 constraints, 1025 triggers.

Repository (2)

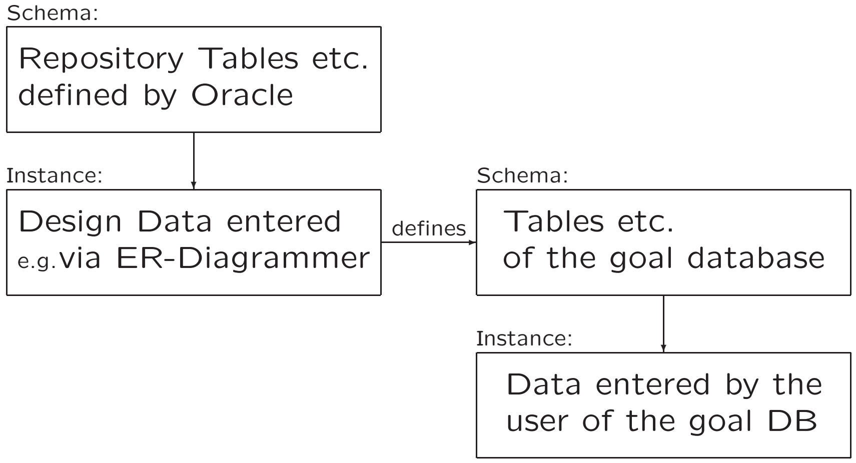
- A repository is similar to a data dictionary (system catalog).

  It also contains meta-data (data about data, e.g. table and column names of another database are data in the repository database).

- Whereas a data dictionary contains only the relational schema, the repository contains all design information, including, e.g. ER-diagrams and application program designs.

  Also, a data dictionary contains meta-data of the database itself. The repository contains information about a different database (that still needs to be constructed).

Repository (3)



Repository (4)

- Design data about several projects (each in several versions) can be stored in the same repository.

  In Oracle Designer, projects are called "Application Systems" (the goal of the project: database + application programs).

- In order to use Oracle Designer, one needs an account on the DB in which the repository is stored.

- The user who has installed the repository under his/her account is called the repository owner.

  He/She can give access to the repository to other users of the same database. This is done with the repository administration utility (RAU) and the repository object navigator (RON), see Appendix C.

Repository (5)

- The different tools are integrated, because they all store and retrieve their information from the common repository.
- The multi-user features are also in part inherited from the underlying database system.

  Longer term locks are managed by flags in the repository tables.
- Oracle Designer is user-extensible: The Repository has a documented application program interface.

  It consists of view definitions and PL/SQL procedures.
  See also below for user-defined properties.

Repository (6)

- E.g. developers can write their own style-checker for ER-diagrams using the information stored in the repository.
- The technical way to learn about Oracle Designer is to understand the database schema ("meta model") defined by Oracle for storing design information in the repository.

  Actually, besides a tutorial and the quite extensive online help, the only documentation shipped with Designer 2.1.2 explains the repository API and the meta model.

Repository (7)

- For the repository, Oracle defined
  - ◇ Element types (such as entity, attribute, relationship end, unique identifier) and for each element type a number of properties.

    There are about 200 different element types.
  - ◇ Association types (i.e. relationships between element types, e.g. "function entity usage")

    There are 39 different association types.
  - ◇ Text types (e.g. description, PL/SQL block).

    There are 38 different text types.

Repository (8)

- The Repository Object Navigator (RON) can be used to view and edit most of the properties of the repository elements.
- Also direct SQL access to the views is possible, updates can be done via the API PL/SQL procedures.
- Tools like the ER-Diagrammer give a nicer interface to a subset of the information that can be accessed with RON.

  Actually, the graphical data (position of entities) cannot be accessed with RON. But one can e.g. edit attributes of entities.

Repository (9)

- The complete meta model is defined under

  `Programs` → `Repository 6i Doc` → `Model Reference`

- E.g., entity types have the following properties:
  - ◇ `OWNING_CONTAINER`: Name of application system (or version), to which this entity belongs.
  - ◇ `NAME`: Name of the entity type (singular form).
  - ◇ `PLURAL`: Plural form of the name (table name).
  - ◇ `SHORT_NAME`: Shorthand for the entity type name (used in generated foreign key column names).

Repository (10)

- Properties of entity types, continued:
  - ◇ `TYPE_OF`: Name of superclass (if this is a subclass).
  - ◇ `INITIAL_VOLUME`: Initial number of entities of this type.
  - ◇ `ANNUAL_GROWTH_RATE`: Expected increase (in %) in the number of entities per year.
  - ◇ `VOLUME`: Average number of entities of this type.
  - ◇ `MAXIMUM_VOLUME`: Maximal number of entities of this type.

Repository (11)

- Properties of entity types, continued:
  - ◇ `DATAWAREHOUSE_TYPE`: “Dimension” or “fact” (only for data warehouse applications).
  - ◇ `DESCRIPTION`: Explanation of the meaning of the entity type (definition).
  - ◇ `NOTES`: Annotation for the database designers.
- All of these properties correspond to fields in the dialog box that contains the entity data.

  This dialog box also permits to specify attributes. “Attribute” is another element type of the meta model with its own set of properties.

Repository (12)

- Standard properties of all element types:
  - ◇ `ID`
  - ◇ `DATE_CREATED`
  - ◇ `CREATED_BY`
  - ◇ `DATE_CHANGED`
  - ◇ `CHANGED_BY`
  - ◇ `NUMBER_OF_TIMES_MODIFIED`
  - ◇ `ELEMENT_TYPE_NAME`
  - ◇ `TYPES`
  - ◇ `CHECKOUT_VERSION`

Repository (13)

- The view CI_ENTITIES has the following columns:
  - ◇ ANNUAL_GROWTH_RATE
  - ◇ CHANGED_BY
  - ◇ CREATED_BY
  - ◇ DATAWAREHOUSE_TYPE
  - ◇ DATE_CHANGED
  - ◇ DATE_CREATED
  - ◇ ELEMENT_TYPE_NAME
  - ◇ ID

Repository (14)

- Columns of CI_ENTITIES, continued:
  - ◇ INITIAL_VOLUME
  - ◇ INSTANTIABLE_FLAG
  - ◇ IRID
  - ◇ IVID
  - ◇ MAXIMUM_VOLUME
  - ◇ NAME
  - ◇ NUMBER_OF_TIMES_MODIFIED
  - ◇ PLURAL

Repository (15)

- Columns of CI_ENTITIES, continued:
  - ◇ SHORT_NAME
  - ◇ SUPERTYPE_REFERENCE
  - ◇ TYPES
  - ◇ USER_DEFINED_PROPERTY_0
  - ◇ USER_DEFINED_PROPERTY_1
  - ◇ ...
  - ◇ USER_DEFINED_PROPERTY_19
  - ◇ VOLUME

Repository (16)

- In Designer 2.1.2, most of the views for element types are selections and projections of the table SDD_ELEMENTS that contains elements of many different types.

  But this table is not part of the published API.

Repository (17)

- The repository is user-extensible: One can add properties to existing element types, and it is also possible to add element types, association types, and text types.

  This is done with the Repository Administration Utility (RAU).

- So if one thinks that there is other important information that should be recorded about entity types (or other design elements), it is possible to extend Oracle Designer in that way.

CASE Tool Advantages (1)

- All documentation in one place.
- Easy modifications. Version Management.
- Graphical editors for various types of diagrams that enforce the syntax of the respective diagram type.

  It is easier to use a graphical editor that knows about ER-constructs than a general graphical editor. In addition, the resulting ER-diagram is guaranteed to be well-formed.

- Information in the repository is structured, and not pure text.

  This makes searches more selective, and allows complex evaluations (e.g. average number of attributes per entity type).

CASE Tool Advantages (2)

- Browsing of the information is possible (hyperlinks).

  E.g. one can click on an entity type in an ER-diagram and gets a dialog box that contains additional information.

- Cross-references are enforced (no broken links).
- Consistency between documents.

  E.g. if the same entity type appears in two diagrams, it has the same attributes: It is stored only once in the repository.

- Different views on the same data: Design documents at different levels of detail.

  The same piece of information has to be entered only once instead of once for every document that contains it.

CASE Tool Advantages (3)

- The translation from an ER-schema to a relational schema is quite tedious if done by hand.

  It is also easy to make mistakes in a manual translation. The consistency between the ER-diagrams and the relational schema is guaranteed if an automatic translation is used. If the two get out of sync the ER-diagrams become misleading and a useless documentation.

- A CASE tool contains knowledge
  - ◇ how software should be developed and
  - ◇ which documentation is needed.

Overview

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.

Entities and Attributes (1)

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

```
INSTRUCTOR

# NAME
* ROOM
○ PHONE
```

- 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:

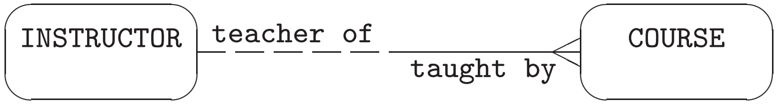
```
INSTRUCTOR

# NAME
* ROOM
○ PHONE

e.g. Brass
```

- 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).

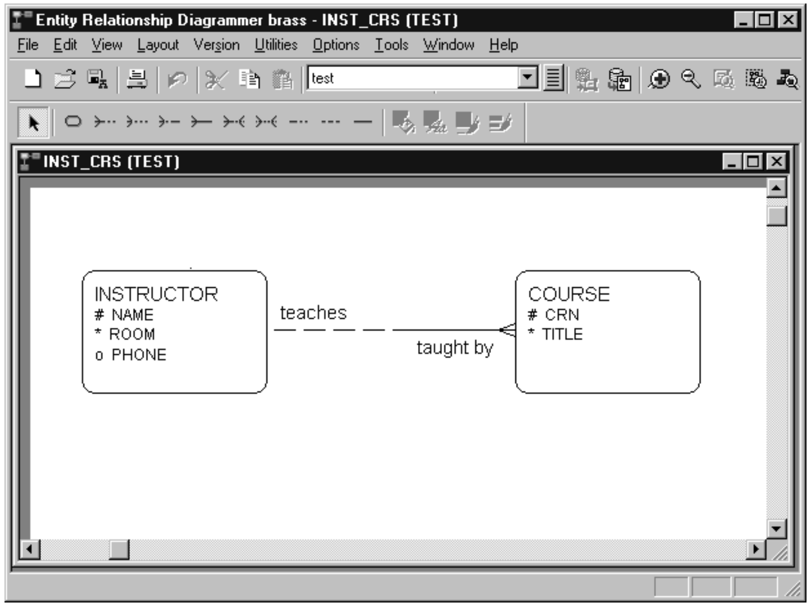
Relationships (2)

- 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.



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.

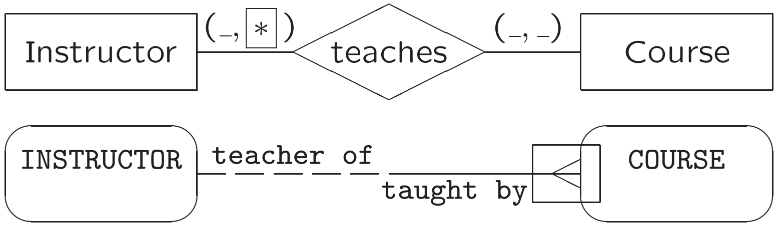
  The maximum cardinality "*" is written on the instructor side (i.e. the "one" side): The (min,max)-cardinalities on the instructor side describe the outgoing edges from a single instructor (number of courses).

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).

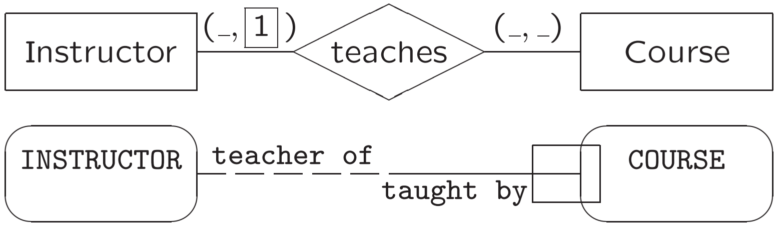
Notation for Cardinalities (4)

- 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:



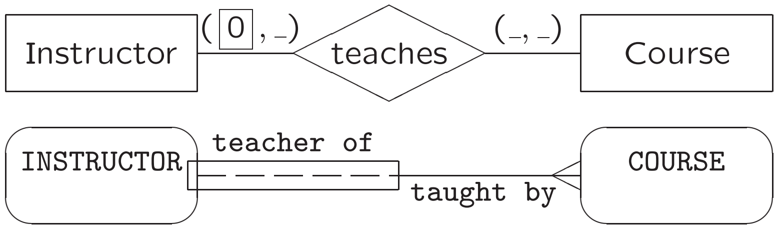
Notation for Cardinalities (5)

- 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:



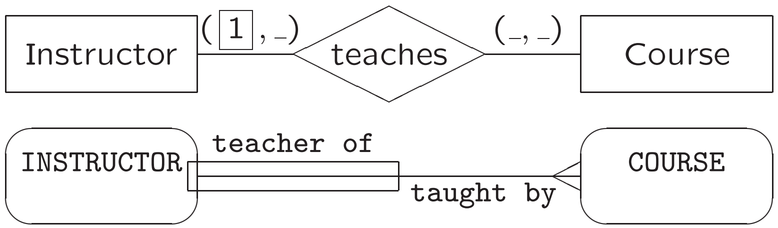
Notation for Cardinalities (6)

- 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).

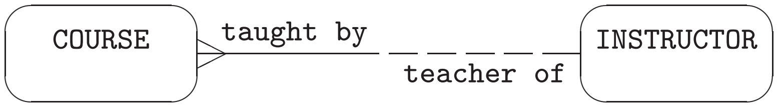
  Keys and other constraints could be validated in the same way.
- Actually, one can even produce a question form and let the user check whether it is true:

  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)":

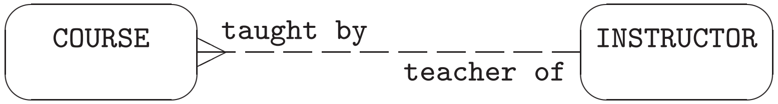
- In Oracle Designer:



Cardinalities (2)

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

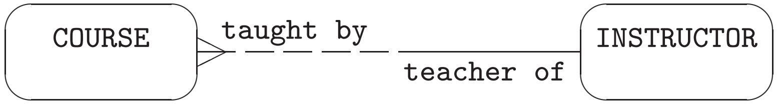
- In this case, a course has not necessarily a teacher assigned.
- In Oracle Designer:



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):

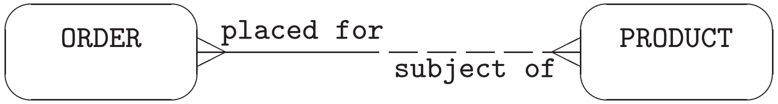
- In Oracle Designer:

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:

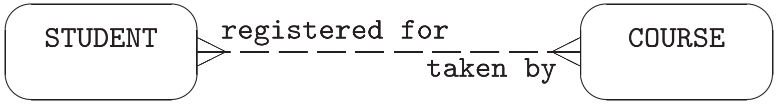


Every purchase order must be for at least one product, but can be for many products. One product can be ordered in many purchase oders. There can be new products that are not yet ordered.

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)

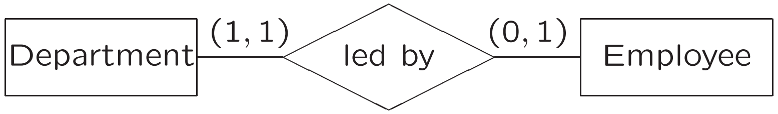
- Many to many (mandatory to mandatory):

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

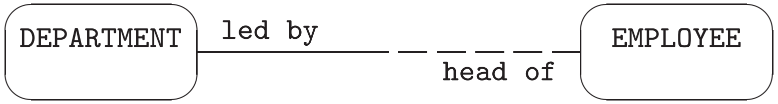
  A student cannot be inserted without a course, and a course cannot be inserted without a student. In general, when one defines cardinalities, one should think about elementary transactions. Which insertions/deletions must happen together such that the cardinality requirements are satisfied at the end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

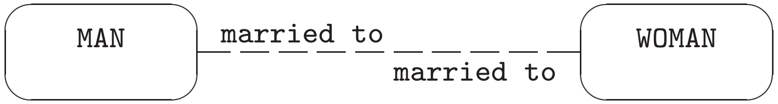


Every department is led by exactly one employee, an employee can be head of at most one department.

Cardinalities (9)

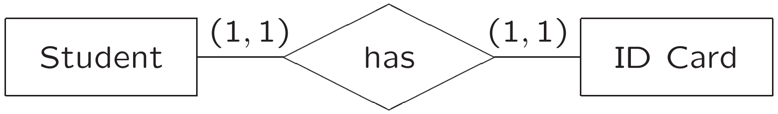
- One to one (optional to optional):

- In Oracle Designer:

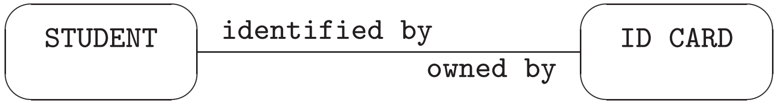


Cardinalities (10)

- One to one (mandatory to mandatory):



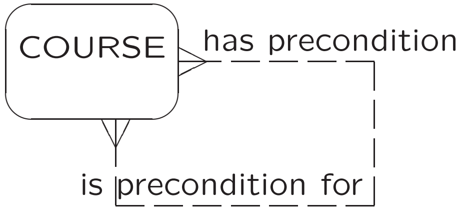
- In Oracle Designer:



- 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

1. Oracle Designer
2. Entities and Relationships
3. Multiple Diagrams for one Schema
4. Attributes, Domains, Advanced Constructs
5. Repository Reports, Rep. Object Navigator

Multiple ER-Diagrams (1)

- Real World Database Schemas are often very large (hundreds to thousands of entity types).
- If one produces only one diagram with all entity types on it, one can use it as a wallpaper.
- It would be very difficult to find something on such a large diagram.
- Even as a means for getting an overview or for orientation purposes it would be useless.

  The only use of such a large diagram is to get an impression of the size and complexity of the database schema.

Multiple ER-Diagrams (2)

- Therefore, the schema must be split into several diagrams.
- Oracle Designer (and similar CASE tools) distinguish between
  - ◇ the global database schema (stored in the repository)
  - ◇ the subset of the information that is shown on a particular diagram.

Multiple ER-Diagrams (3)

- The same entity type can be shown in more than one diagram.

  Since the entity types are connected by relationships, and often form a connected graph, it is necessary to duplicate nodes (entity types) if one wants to split the diagram and still show all edges (relationships).
- Also the same relationship can be shown on more than one diagram, but this happens less frequently.
- It is possible that entity types or relationships in the repository are not shown in any diagram.

  This is probably only a temporary condition or may even be an error (garbage left over from a deleted diagram).

Multiple ER-Diagrams (4)

- When using the ER-Diagrammer, one must distinguish between
  - ◇ Creating a new entity type: This will automatically insert the entity type into the repository.
  - ◇ Including an entity type that already exists in the repository into a diagram.
- Entity type names must be unique within an application system (global schema in the repository).

  If one tries to create an entity type with the same name as one that already exists in the repository, one gets an error message.

Multiple ER-Diagrams (5)

- Relationships must be uniquely identified by the two entity types and the "to" and "from" names.

  Again, if one tries to re-create a relationship that still exists in the repository, one gets an error message.
- When deleting an entity type or a relationship, one must distinguish between
  - ◇ deleting it only from a single diagram,
  - ◇ deleting it from the diagram and the repository.

    From time to time, one should search with the Repository Object Navigator or the Repository Reports Utility for entity types and relationships that were deleted from the diagrams but still exist in the repository.

Multiple ER-Diagrams (6)

- Changes to an entity type done in one diagram will be semi-automatically reflected in other diagrams.
  - ◇ If an entity or relationship is changed in the repository, outdated versions still contained in other diagrams will be marked with a red bullet.
  - ◇ There is a special command to update entities or relationships on the diagrams from the repository.

Overview

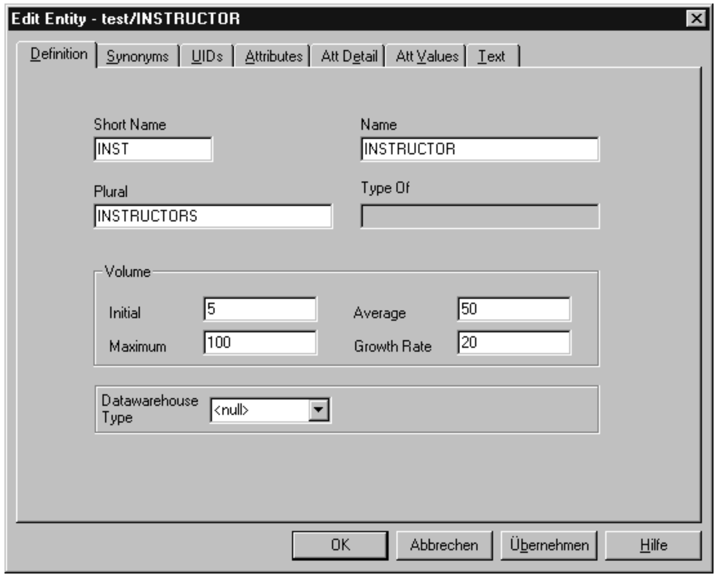
Entity Properties (1)

- 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).



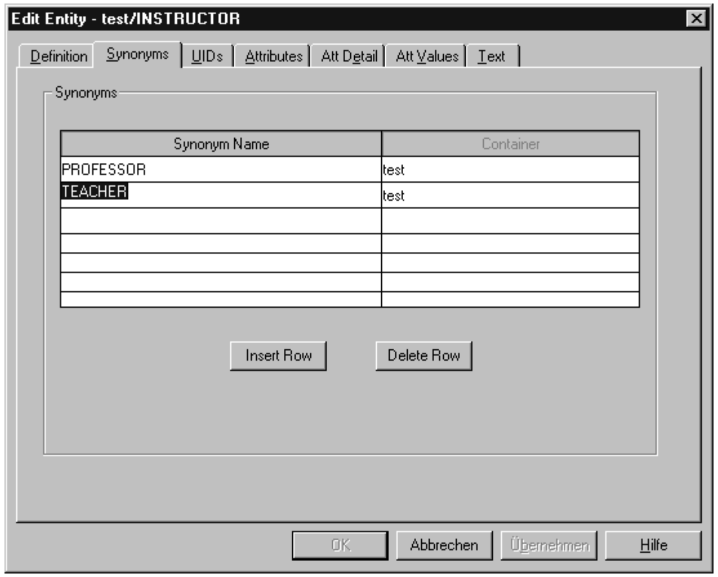
Entity Properties (4)

“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).

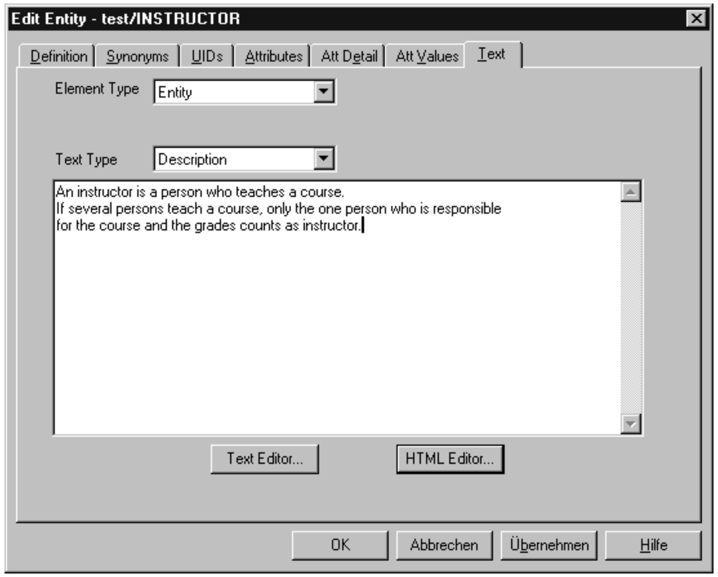
  “Fact” vs. “Dimension” tables (see below).



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.

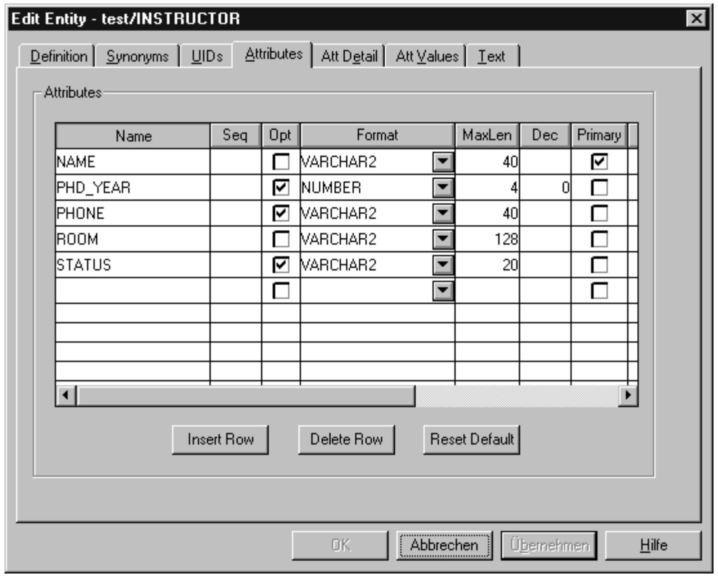


Entity Properties (8)

"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.



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)

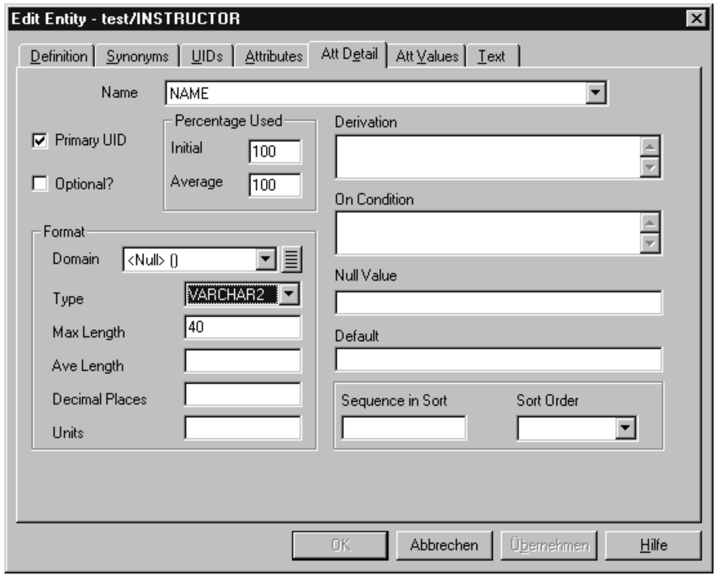
- Possible attribute formats (data types) are `CHAR`, `DATE`, `IMAGE`, `INTEGER`, `MONEY`, `NUMBER`, `PHOTOGRAPH`, `SOUND`, `TEXT`, `TIME`, `TIMESTAMP`, `VARCHAR2`, `VIDEO`.
- Not all of these data types exist in Oracle or all the other supported goal systems.
- It is a task of the logical design phase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



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)

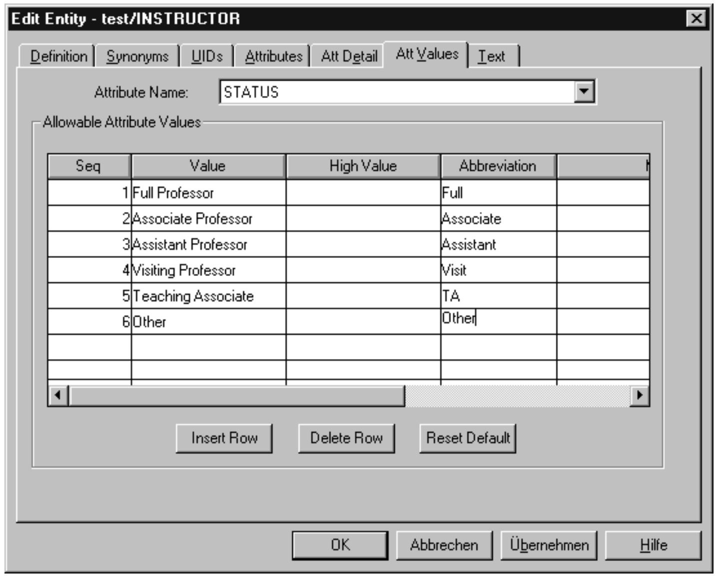
- Information on the “Attribute Detail” page:
  - ◇ Physical design information: average length and percentage of entities having a non-null value.
  - ◇ Units for the attribute (e.g. “kg”, “cm”, “in”).
  - ◇ A derivation formula/algorithm if this attribute is derived.
  - ◇ A condition when this attribute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



Entity Properties (18)

“Attribute Values” Page:

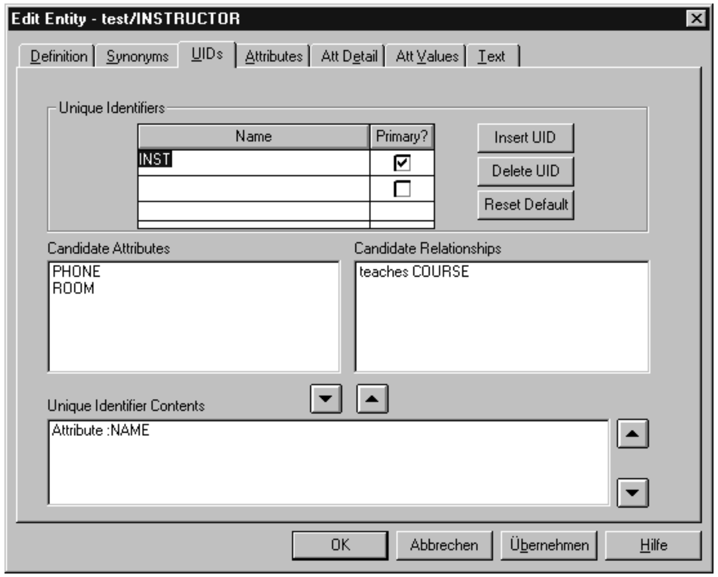
- On this page, restrictions for the values of an attribute can be defined (e.g. for “enumeration type” attributes).
- One can define all possible values of an attribute:
  - ◇ Value
  - ◇ Sequence number

    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.



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).

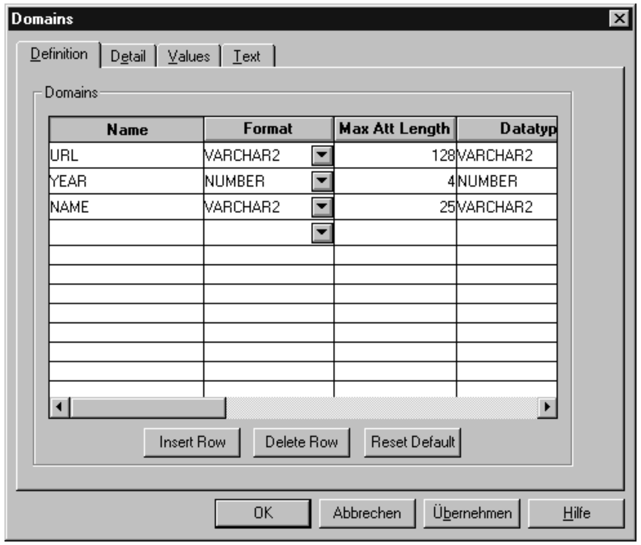
  This is not implemented in Oracle 8. But when domains are defined in the Designer, they might be partially mapped to SQL domains in other DBMS. Oracle 8 has PL/SQL types which could also be used. But for consistent schema changes, it is already helpful that they are supported in the Designer.

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.

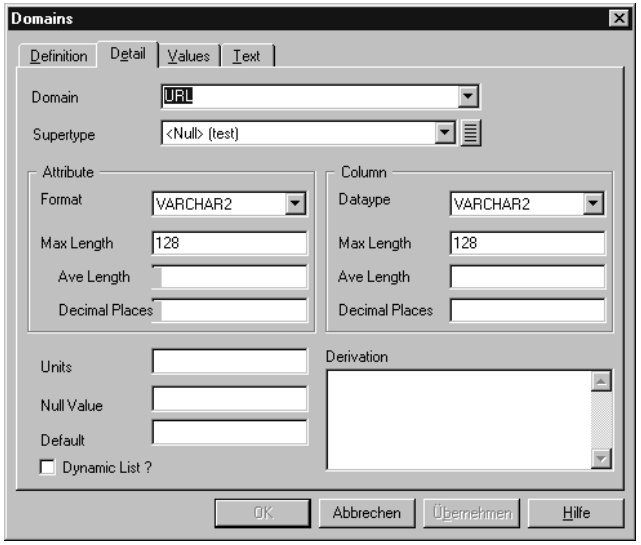
  E.g. names of length 10, 20, 40, descriptions of size 2000, email/URL of size 250, ZIP codes, SSN, boolean values, etc. Selecting from a set of predefined standard domains can be done faster than considering every attribute in isolation. In some projects, only a "domain administrator" is allowed to create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

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



Domains (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 (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.

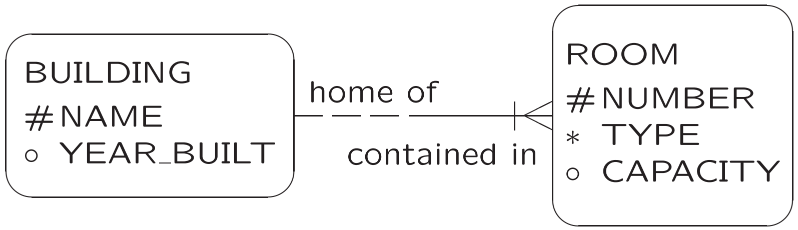
  This makes it easy to change the possible values of the enumeration type later: One can simply insert a new value into the lookup table.

- 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.

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.

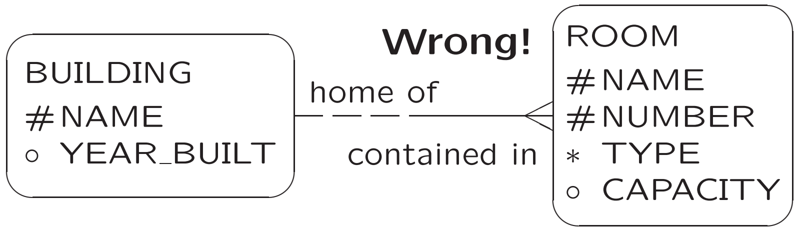
  The building name in addition will be a foreign key that references the `BUILDINGS` table.

- 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).

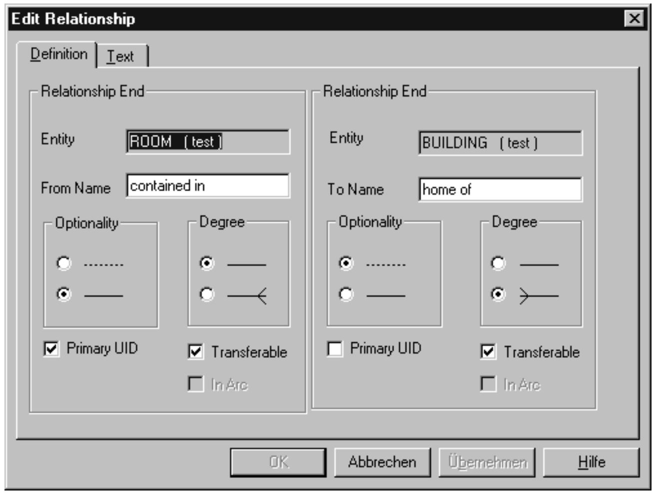
- At least for primary keys, the participation in the relationship must be mandatory.

  Primary key attributes cannot be null.

Weak Entity Types (7)

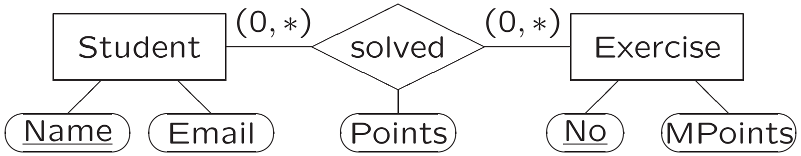
- There are two places to specify that a relationship contributes to the identification of the entity:
  - ◇ In the entity definition, tab UIDs.
  - ◇ In the relationship definition.

    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.



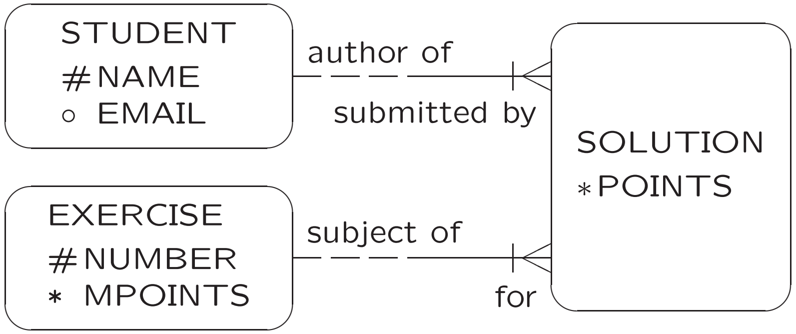
Association Entity Types (1)

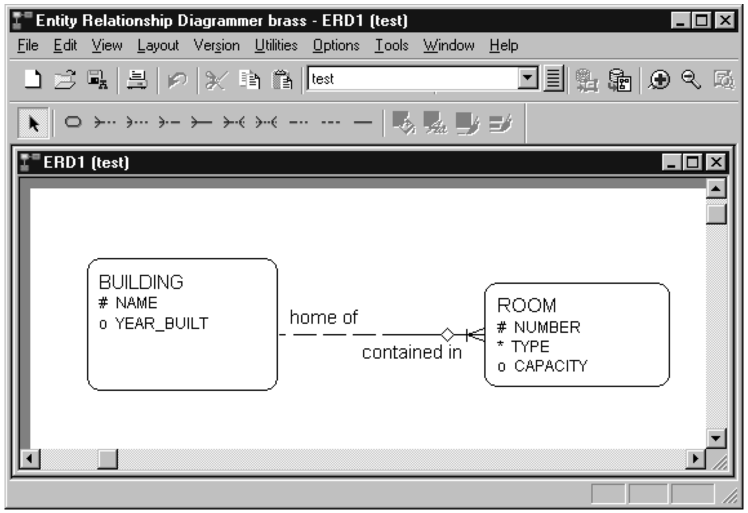
- Weak entity types can have more than one parent.
- Weak entity types with two (or more) parent types are sometimes called "association entity types", because they are similar to a kind of relationship between the parent entity types.
- E.g. suppose that we need a relationship attribute:



Association Entity Types (2)

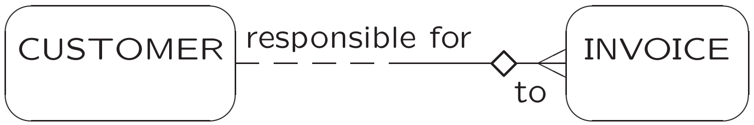
- Oracle Designer does not support relationship attributes. However, one can turn the relationship into an association entity type:





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.

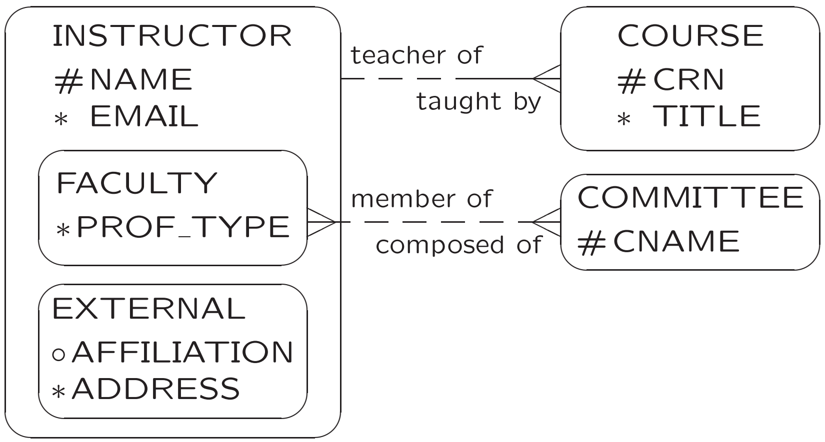
- Non-Updatability is a simple kind of dynamic constraint, which refers not to single database states as a normal static constraint, but to pairs of DB states.

  Another example is "Salaries cannot decrease."

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 you partial specialization, but 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.

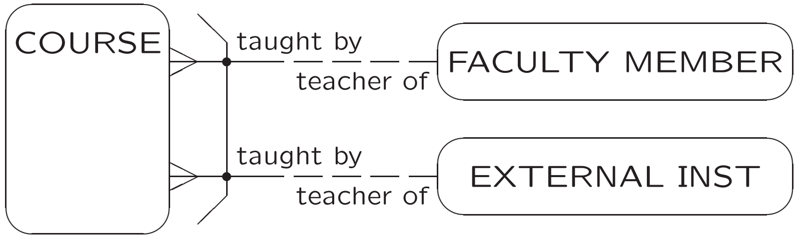
  Constructs like specialization reduce the need for constraints.

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".

Overview

1. Oracle Designer
2. Entities and Relationships
3. Multiple Diagrams for one Schema
4. Attributes, Domains, Advanced Constructs
5. Repository Reports, Rep. Object Navigator

Repository Reports (1)

- Design documentation can be printed from the information gathered with the ER-diagrammer.
- Diagrams can be printed with the ER-Diagrammer.
- Textual documentation can be generated with the "Repository Reports Utility".
- The goal is that Oracle Designer manages all design information (in the repository).

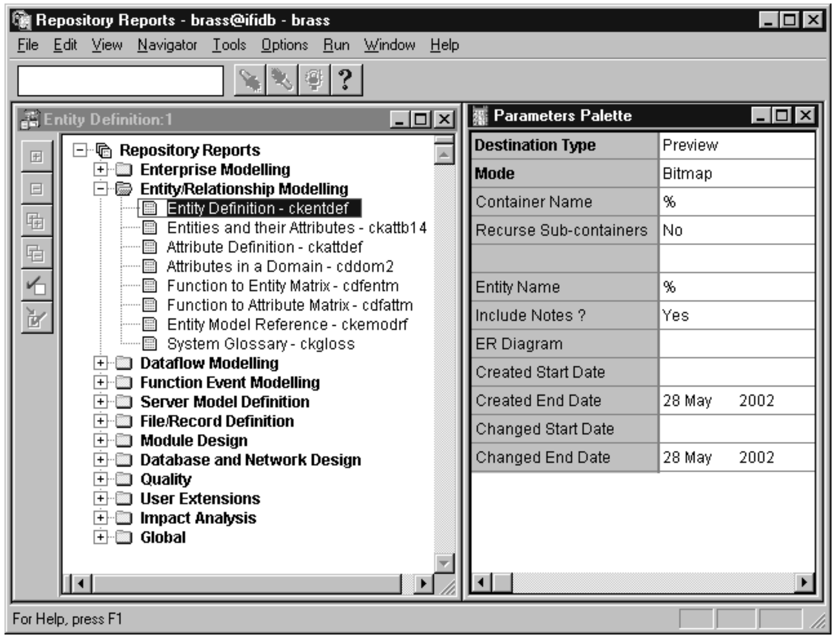
  Ideally, one writes no documents with Word or LaTeX, but enters all information into the repository, and then prints all documentation with the Repository Reports Utility.

Repository Reports (2)

- There are predefined reports at different level of detail. Many reports have parameters, e.g. whether to include all entities or only a specified subset.
- Reports can be shown on screen, written to a file, or directly printed. Possible output formats include HTML and PDF.
- Oracle applies its own "Oracle Developer Reports" Report Generator. One can use this tool to develop one's own reports.

Repository Reports (3)

- Of course, automatically generated documentation has a very regular structure and is very dry to read, but it is easy to look up the information one needs.
- Also, it depends on the designers, how much textual comments they store about entities, relationships, and attributes.
- Sometimes, it might be easier to find the information in the repository with Oracle Designer, but customers expect the usual printed documentation.



Repository Reports (5)

- Reports listed under Entity/Relationship Modelling:
  - ◇ Entity Definition

    One page per entity (with attributes).
  - ◇ Entities and their Attributes

    Overview table, one line per attribute.
  - ◇ Attribute Definition

    Half a page per attribute, attributes are sorted by entity types, a new page is started for a new entity type.
  - ◇ Attributes in Domain

    Table with one row per attribute in a domain, ordered by domain.

Repository Reports (6)

- Entity/Relationship Modelling Reports, continued:
  - ◇ Entity Model Reference

    One (or more) pages per Entity, includes use in business functions.
  - ◇ Function to Entity Matrix

    CRUD information for all entity types accessed by a function.
  - ◇ Function to Attribute Matrix

    Detailed CRUD information.
  - ◇ System Glossary

    Descriptions of entity types.

Repository Reports (7)

- Reports listed under “Quality” do style checks or find certain errors that can exist during the design process but should be removed before the relational schema is generated:
  - ◇ Entity Completeness Checks

    This lists entities that have no attributes, no description, no keys (UIDs), or that are not used by any function.
  - ◇ Quality Checking of Relationships

    This lists “uncommon” relationships, namely many-to-many relationships and one-to-one relationships. Some designers believe that they should manually translate many-to-many relationships into association entities. I do not share this view.

Rep. Object Navigator (1)

- The global schema (all defined entities and relationships) can only be seen in the Repository Object Navigator (RON).

  Each diagram allows only to see and edit the entities and relationships contained in that diagram.

- The global schema is used for the translation into the relational model, and for printing reports.

  It happens that entity types remain in the repository that were already deleted on the diagrams. These can be found with RON.

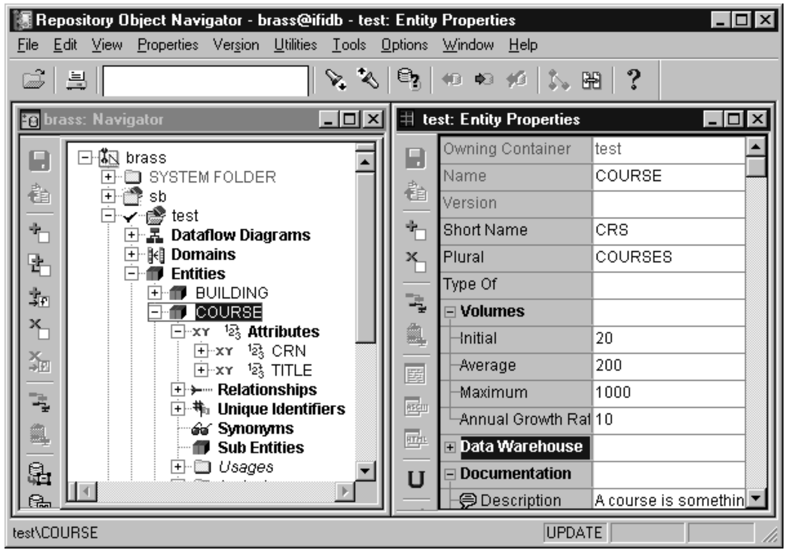
Rep. Object Navigator (2)

- RON has a navigator window (on the left) and a property palette (on the right).

  The navigator window has the standard interface for browsing tree-structured data (expand/collapse subtree). Subtrees correspond to directories. There are a lot of options for selecting what to display (one can define filter conditions).

- Objects can be created or deleted and properties can be changed also in RON.

  One must use "Save uncommitted data" in the file menu to save the changes to the repository. If one has changed the data with another tool (e.g. the ER-Diagrammer) while RON was open, one must use "requery all" to reload the information from the repository.



Rep. Object Navigator (4)

- In Designer 2.1.2, the main categories were:
  - ◇ Reference Data Definition (contains domains)
  - ◇ Enterprise Modelling
  - ◇ Entity/Relationship Modelling (ER-diagrams)
  - ◇ Dataflow Modelling
  - ◇ Function Event Modelling
  - ◇ Type Modelling
  - ◇ C++ Object Generation
  - ◇ Server Model Definition (contains tables)

Rep. Object Navigator (5)

- Main categories in Designer 2.1.2 RON, continued:
  - ◇ File/Record Definition
  - ◇ Module Design
  - ◇ Database and Network Design
  - ◇ Sets
- It seems that in Designer 6i, many of the element types of the meta model can appear, but only non-empty containers are shown by default.

  To see all containers, go to "Navigator→Show/Hide", select "Always" for "Objects". Then quit RON and start it again.

Rep. Object Navigator (6)

- Designer 6i has a lot of sections/folders.

  Assumptions, Business Functions, Business Terminology, Business Units, Cluster Definitions, Communities, Critical Success Factors, Data Items, Dataflow Diagrams, Datastores, Diagrams, Documents, Domains, Entities, Entity-Relationship-Diagrams, Externals, Files, Function Hierarchy Diagrams, Java Class Definitions, Key Performance Indicators, Languages, Locations, Materialized View Definitions, Matrix Diagrams, Module Diagrams, Modules, Nodes, Non-Persistent Queues, ODD Diagrams, Objectives, Oracle Collection Types, Oracle Databases, Oracle Object Types, Other Databases, PL/SQL Definitions, Persistent Queues, Preference Sets, Preferences, Problems, Process Events, Process Models, Queue Table Definitions, Record Files, Records, Reusable Lists of Values, Reusable Module Components, Sequence Definitions, Server Model Diagrams, Source Files, Storage Definitions, Synonyms, Table Definitions, Transformation Mapping Sets, User Defined Sets, View Definitions, Usages, Inclusions.

Rep. Object Navigator (7)

- The section “Entities” lists all defined entity types.
- If one expands an entity, one gets:
  - ◇ Attributes
  - ◇ Relationships
  - ◇ Unique Identifiers
  - ◇ Synonyms
  - ◇ Sub Entities (subclasses)
  - ◇ Usages (e.g. links to business functions, table)
  - ◇ Inclusions (e.g. links to diagrams)

Rep. Object Navigator (8)

- It is possible to define “Filters”, i.e. simple queries that determine what is actually shown.

  E.g. one can select the “Entities” section and then the menu item “Navigator→Filter”. Select the radio button “Data”, not “Headings”. Then one can enter conditions for the name, plural, short name, and supertype of the entity type, as well as conditions for the data volumes (Initial, Average, Maximum, Annual Groth Rate), and the datawarehourse type. The comparison for the text fields is done with “`LIKE`”, so one can use “`%`” and “`_`” as usual.

- One can also specify the sort order e.g. within the “Entities” section. and select which properties are shown in the proterty palette.