Conceptual Schema & Relational Database Design, 2nd ed: Updates and Errata

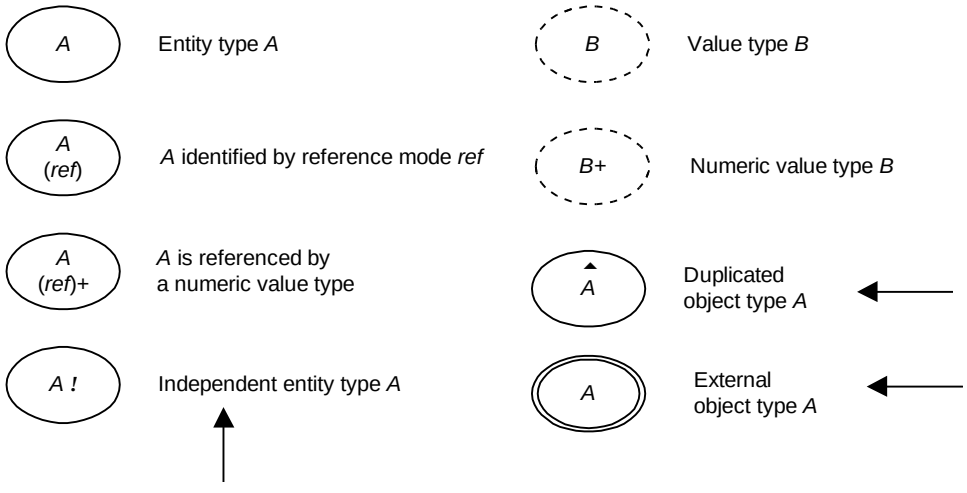
by Dr. Terry Halpin, BSc, DipEd, BA, MLitStud, PhD
Director of Database Strategy, Visio Corporation

The second edition of Conceptual Schema and Relational Database Design *(Prentice Hall Australia 1995), uses a few ORM notations that differ from those implemented in Visio's InfoModeler, and also contains some errata. These differences and errata are summarized in this document, which may be freely copied. If you find other errata, please email me the details at terryh@visio.com, and I will update this document. Any later editions of the book will match the ORM notation more closely to that of Visio modeling tools.*

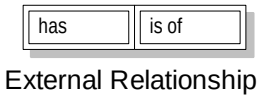
Comparison with InfoModeler notation

The following symbols are used in InfoModeler for object types and reference modes. In addition, InfoModeler lets you flag value types during textual input by appending empty parentheses. For example, you might enter the following fact type in the fact editor: Employee (empnr) has Surname(). The tool will still draw the Surname value type as a named, dotted ellipse (without parentheses).

Numbered arrows point out three differences from the book notation. The term "independent" replaces "lazy" [1]. As an aside, I'm still not happy with this descriptor– if you can think of a better one, let me know. If an object type is duplicated anywhere in a model (same page or different page) this is shown by "▲" [2]. In the book I used "Δ" to mean duplication on a different page, and a double ellipse to mean duplication on the same page. The double ellipse is now used to mean the object type has been imported from an external model [3].

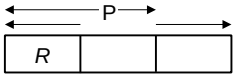


If a predicate is imported from an external model, InfoModeler uses double lines for the role boxes (see opposite).

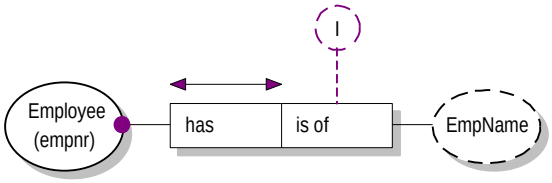


Various styles can be applied to display predicates. By default, InfoModeler uses shading, as shown. For an *n*-ary predicate ($n > 0$), InfoModeler allows *n* predicate readings (one starting at each role), as well as *n* role names to be entered in the predicate properties sheet. For binaries, the user may choose to display the forward reading only, the inverse reading only, or both (as shown).

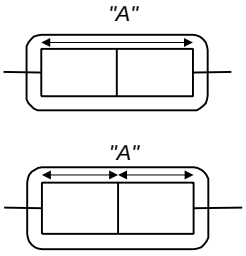
If a predicate has one or more internal uniqueness constraints, InfoModeler allows one of these to be designated as primary, by superimposing "P". When used for fact types, this logical annotation is not a conceptual issue, and is only for mapping purposes (e.g. to determine the primary key of a relational table).



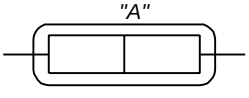
As a physical annotation, InfoModeler allows indexes to be declared using a circled "I" attached by a line to the relevant role(s). This is not a conceptual issue, and is used only for mapping. In the example shown opposite, the index constraint causes an index to be created for EmpName in the Employee table resulting from relational mapping.



In the book, disjunctive external uniqueness is indicated by appending a "%" to the external uniqueness symbol (Ⓤ or Ⓟ). InfoModeler does not append the "%" sign. It is not really needed, since the disjunctive nature can be deduced from the associated mandatory role pattern.

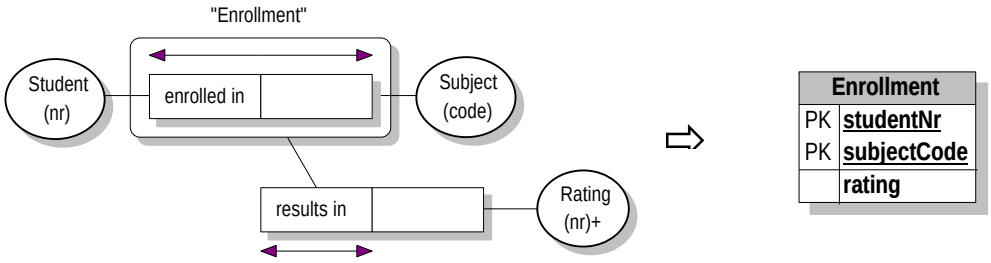


For nested object types, InfoModeler always displays the uniqueness constraint(s) on the objectified association (see opposite). In the book, it was allowed to hide the constraint (see below). In this case, a spanning uniqueness constraint was assumed, so the below case was taken to be equivalent to the top case opposite.

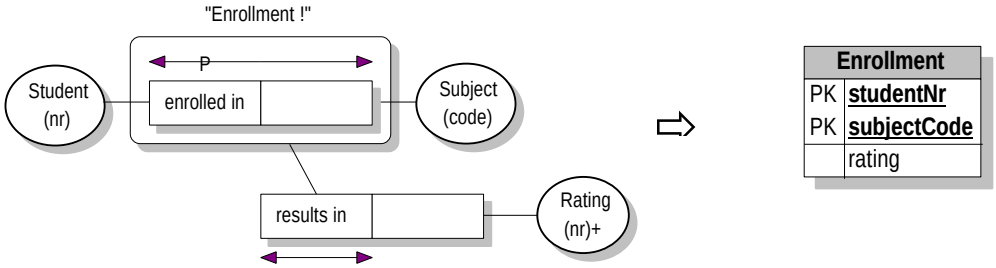


In the book, independent (lazy) entity types had to have "!" appended to their names if they were simply identified or co-referenced, but not if they were objectified associations.

InfoModeler requires independence to be explicitly declared for all three cases. This is an important difference, because different mapping results. For example, in the following example, Enrollment has not been declared independent, so rating is mandatory in the mapped table. InfoModeler displays table columns vertically, with mandatory columns in bold, and uses "PK" for the primary key (as well as underlining). In the book, this table scheme is shown as Enrollment (studentNr, subjectCode, rating).

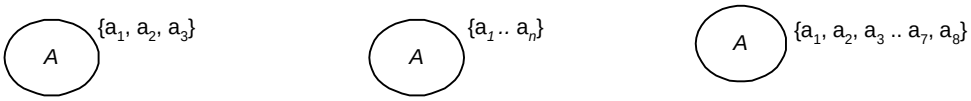


In the book, the above Enrollment is assumed to be independent by default, leading to an optional rating column. To achieve that kind of mapping in InfoModeler, Enrollment must be explicitly declared as independent (see below). InfoModeler issues a warning in such cases so that the modeler can make a definite decision about what is required. Here we know the rating column is optional, because it is not in bold. In the book, optional columns are shown in square brackets, and are otherwise mandatory, so the table scheme is: Enrollment(studentNr, subjectCode, [rating]). During mapping, InfoModeler deduces that the uniqueness constraint on the enrollment association provides its primary reference, and redisplays it with the "P" as shown. You can turn off the "P" display by unchecking this option on the constraint's properties sheet.

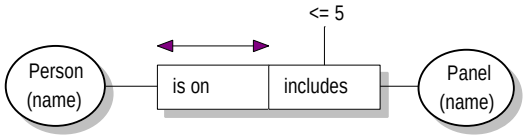


Disjunctive mandatory role constraints are usually depicted by a black dot where the roles connect to their object type (see left option below). An alternative and slightly more general notation is to connect "⊙" to the relevant roles (see right option below). In the book, the right-hand option connected the "⊙" to the role arcs instead.

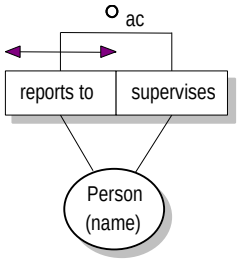
In the book, value-list constraints were restricted to a single list or a single range (see left two cases below). InfoModeler also allows combinations of these cases (e.g. see right-most case below).



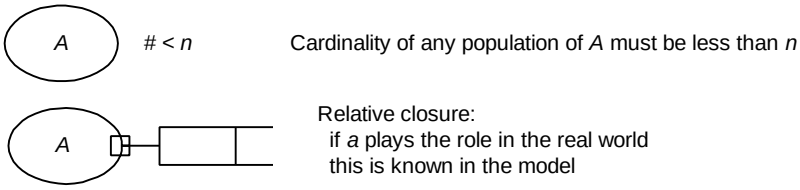
InfoModeler always uses "<= *n*" to display frequency constraints of "at most *n*", and connects the constraint by a line to the relevant role(s), as shown opposite. The book allowed this notation as well as the alternative "1 – *n*" notation (e.g. "1 – 5" for the case shown), and did not show connectors.



For ring constraints, InfoModeler always shows the role connectors, even if the roles are contiguous (see opposite). In the book, the role connectors are usually omitted.



Two constraints mentioned in the book that are not supported in InfoModeler are object cardinality and relative closure constraints (see below).



Subtype definitions and extra rules may be entered textually in InfoModeler, but are not mapped. Some of the more advanced constraints may be specified but are not fully mapped (e.g. join constraints and acyclicity). Later releases of Visio products should eventually provide full support for these additional features. InfoModeler allows you to specify annotations for object-relational structures (e.g. row-types, sets, lists, bags) for logical mapping, but since these notations are likely to change, they are not discussed here.

In relational table schemes, InfoModeler indicates uniqueness constraints other than the primary key by annotating columns with "U*n*", where *n* is a number for the relevant constraint. Foreign key columns are annotated by "FK". A choice of styles is available for

displaying these schemes. More complicated constraints on table schemes (e.g. exclusion constraints) are captured as DDL code rather than displayed graphically.

To improve verbalization of constraints in InfoModeler, I introduced the use of a *hyphen* "-" to bind an adjectival phrase to its object term. For example, the uniqueness constraint on the left role of "Person has favorite- Color" verbalizes as "**each** Person has **at most one** favorite Color"; without the hyphen, the verbalization defaults to the awkward "**each** Person has favorite **at most one** Color".

Errata

Page	Revision
4	Delete the three table rows at top of page; Append to bottom of page: " provides some historical background. The final section of the chapter includes a summary and suggestions for further reading."
6	Paragraph 3, the last sentence should read: The "(0,m)" next to Country means that a movie may be exported to zero or many countries. The "(0,m)" next to Movie means that a country may import zero or many movies.
20	Replace "stored fact types" by "base fact types" (meaning "primitive fact types"). Fact types are either base or derived (this reinforces the idea already stated on p. 22 that derived facts may or may not be stored). *This replacement applies throughout Section 2.2.*
53	Paragraph 2, line 1: delete "a " from "a an extract".
56	Figure 3.5: change "New york" to "New York"
61	Last line: change "kgvalue" to "kgValue".
85	Figure 4.13: change caption to: "A one to one association". Change "America" to "United States".
102	Figure 4.37: change predicate to "... is placed in ... at ... ".
131	Figure 5.12: In right hand alternative, connect constraint to role boxes, not to role-object links.
152	Figure 5.27: change refmode of bottom Length from "(m)*" to "(mm)*".
174	Figure 6.9: change "pop r1" to "pop(r1)".
184	Question 2 Figure: Add arrow head to top of C6 subset constraint.

214 Figure 7.2: change "Clty" to "City".

224 Figure 7.12: uniqueness constraint lines should be solid.

248-50 Renumber Figure 8.5 to figure 8.5(a), and change all references to it accordingly.

Renumber Figure 8.6 to figure 8.5(b), and change all references to it accordingly.

253 Figure 8.8: append "+" to (kg)".

275 Question 16 figure: shorten uniqueness constraint on the ternary, to span just the first two roles.

287 Figure 8.41: Roles played by Student and Employee should be mandatory.

313 Figure 8.74: change "month" to "monthcode".

326q Figure 9.5 caption: change "Ucs" to "UCs".

332 Figure 9.15: Hyphens can greatly improve fact and constraint verbalization, e.g. change "has driver A" to "… has … -A", and "has first" to "has first-". However, hyphen usage is not discussed in this edition of the book.

344 Paragraph 1, last 3 lines: Laziness (independence) now needs to be explicitly declared (see page 2 of this document). There are several aspects of the book that need to be updated in this regard.

361 Figure 9.40, Step 1.2: insert "non-functional, " after "if *n* ".

375 InvoiceLine table scheme: Add a note that this scheme is in 1NF only, because of its implied FD: itemcode → title.

380 Figure 9.58: PaperSlot refmode: change "Slot#)" to "(slot#)".

381 Figure 9.59: PaperSlot refmode: change "Slot#)" to "(slot#)".

382 Figure 9.60: change "Person.afflition" to "Person.affiliation" { insert "i" }.

387 Paragraph 2, line 2: the two tables should be: *R1* (emp#, sex, birthdate, job); *R2* (job, salary).

403 Paragraph 1, line 7: change "performs" to "perform".

404 Paragraph 2, line 3: insert "play" after "may".

Paragraph 3, line 2: change "an object- " to "a".

452 Figure C.8: reverse the last two rows; change caption to: "asc" is assumed by default.

Paragraph 1: change discussion to indicate that “asc” is always assumed by default (options are not sticky).

456 Last 2 SQL statements: delete the implied condition “Department.deptcode = Employee.deptcode”.

458 SQL syntax summary: enclose where-clause and order-by-clause in square brackets.

462 Change: **select** Name **from** Person –
where Name **between** ‘Bob’ **and** ‘Fred’
to: **select** firstname **from** Person
where firstname **between** ‘Bob’ **and** ‘Fred’

479 Last paragraph, line 3: change “ration” to “ratio”

487-8 For the population shown in the figure C.19, the SQL query at the bottom of page 487 should return the null set, not the two rows shown. On page 488, the last line of the first paragraph should indicate 3 (not 2) subjects.

The above is the minimal correction. As a harder but more instructive example that does return the two rows shown, change the query to:

List student numbers of those students who study *all* the subjects *worth over 5 credit points*.

The relational algebra formulation now becomes:

(Result | Subject **where** credit > 5) *[*student#, subjcode*]*
÷ (Subject **where** credit > 5 *[*subjcode*]*)

And the SQL becomes:

```
select “student#”
from   Result natural join Subject
where credit > 5
group by “student#”
having count(*) =
     (select count(*) from Subject
      where credit > 5)
```

If we make this change, the SQL code on page 488 should also be changed by adding the credit > 5 condition to each subquery, or by replacing the whole code with:

```
select “student#”
from   Result natural join Subject
where credit > 5
group by “student#”
having count(*) =
     (select count (distinct subjcode) from Subject
      where credit > 5)
```

488 SQL syntax summary: add final “]”.

491 Paragraph 4, line 2: change “one of tools” to “one tool”.

501 Exercise 6.2, Answer 3: Invoice should be connected to its role box in InvoiceLine.

502 Exercise 6.4, Answer 1: Add UC to role of DriversLicense.

510 Answer 9(a): add predicate text “enrolled in”, “… on … got … ” to diagram.
