

Beyond Referential Attributes

Marc J. Balcer

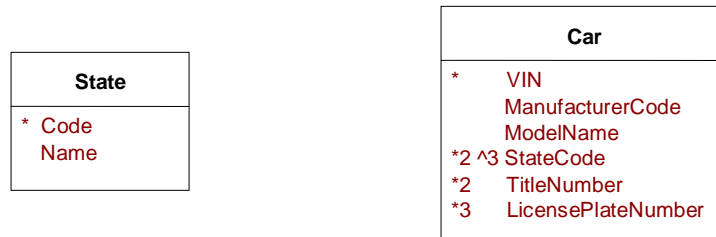
Traditional Shlaer-Mellor information modeling (1) (2) borrows heavily from relational data modeling. Every class is required to have an identifier and every relationship is formalized by copying the identifying attributes into the related object as referential attributes. As in relational data modeling, the selection and placement of referential attributes is used to express certain types of constraints.

Because referential attributes are so tightly coupled to a relational data model, they have been routinely vilified and eliminated from many practitioners' models. Models in *Executable UML* (3) generally did not use referential attributes. Yet there remain cases where referential attributes seem to be the only means of formalizing certain types of constraints.

After understanding the semantics underlying identifiers and relationship formalization I will present an alternative notation based upon these semantics and show how it can be used to solve the problems that seem to require referential attributes.

Identifiers

In traditional Shlaer-Mellor notation identifiers are shown by marking the attributes either with stars



or with {I} tags



These simple identifiers mean, “In the whole population of the class, no two instances of the class may have the same values of the identifying attribute or identifying attribute set.” Identifiers are really constraints, and can also be expressed in the Object Constraint Language (OCL).

context State **inv:** State.allInstances() -> forall(p1,p2 | p1<>p2 **implies** p1.Code <> p2.Code)

An alternative notation for expressing an identifier is to place the identifying constraints into the constraint compartment on the class box. Use the word **unique** to identify a uniqueness (identifier) constraint:

State
Code Name
constraints {unique Code}

This notation, of course, supports the notion of multiple identifiers. If, for example, you want to state that the name of a state also must be unique, simply list it as a second uniqueness constraint:

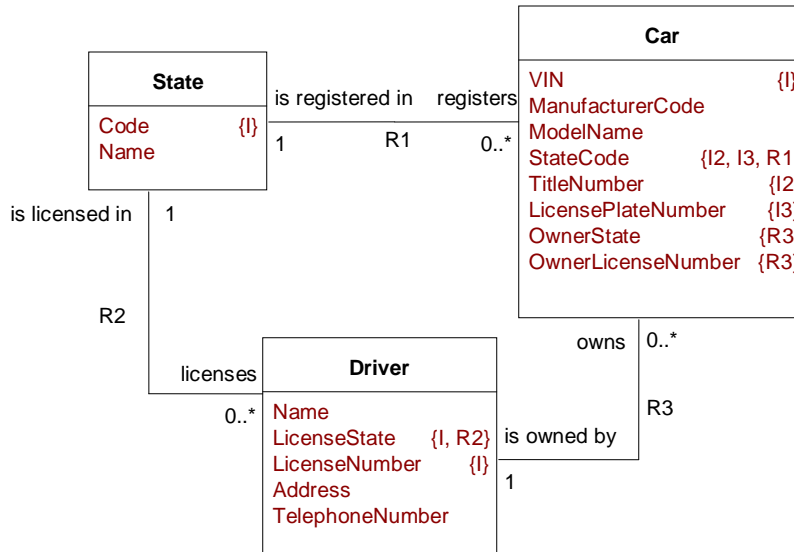
State
Code Name
constraints {unique Code} {unique Name}

A multi-attribute identifier simply lists all of the attributes that, taken together, must be unique.

Car Model
ManufacturerCode ModelName
constraints {unique ManufacturerCode + ModelName}

Referential Attributes

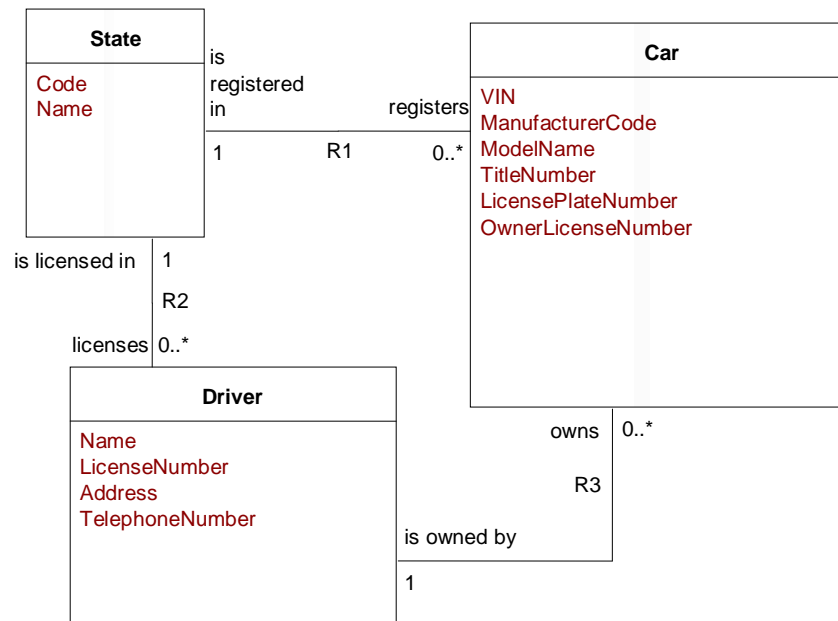
Traditional Shlaer-Mellor notation requires that every relationship—whether an association or a generalization—be formalized by copying the identifier of the “one” object into the “many” as a referential attribute.



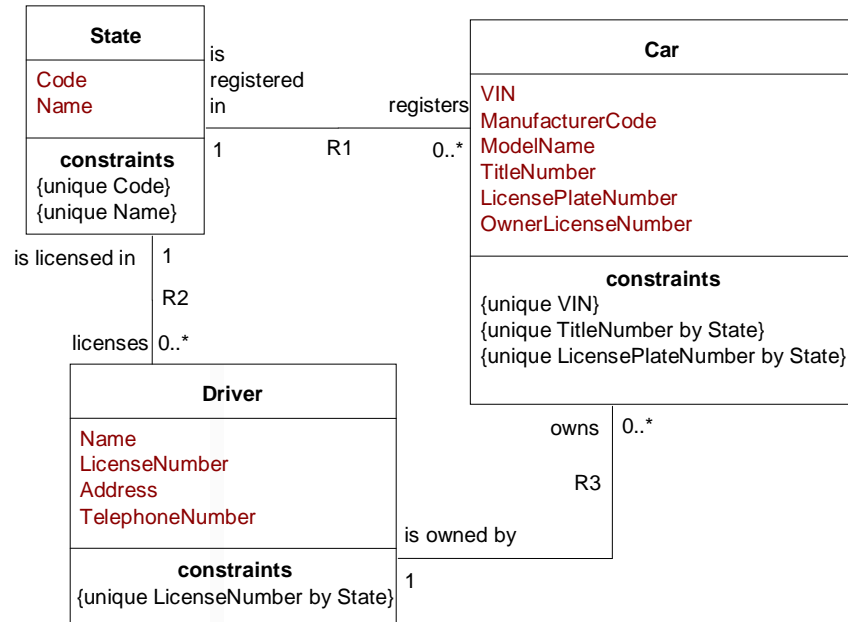
Relationships without Referential Attributes

While referential attributes correctly implement the relational model of data, in cases of unconstrained associations and generalizations, this process of copying identifiers to make referential attributes seems almost mechanical and even may violate the notion of “one fact in one place”—namely that a model should not duplicate information and should say the most with the least content.

By this thinking, each diagram element represents one independent fact. The presence of an attribute means that there is some data characteristic of the class that is meaningful to the domain. The presence of a line between two classes means that there is some association between those two classes. Removing the referential attributes does not take away any of these facts.



Applying the same kinds of constraints, we can complete model of the State, Car, and Driver:

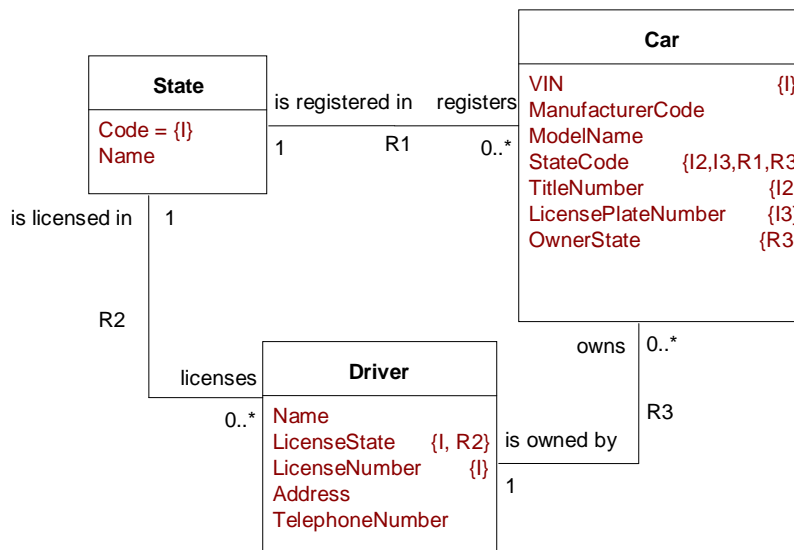


Constrained Relationships

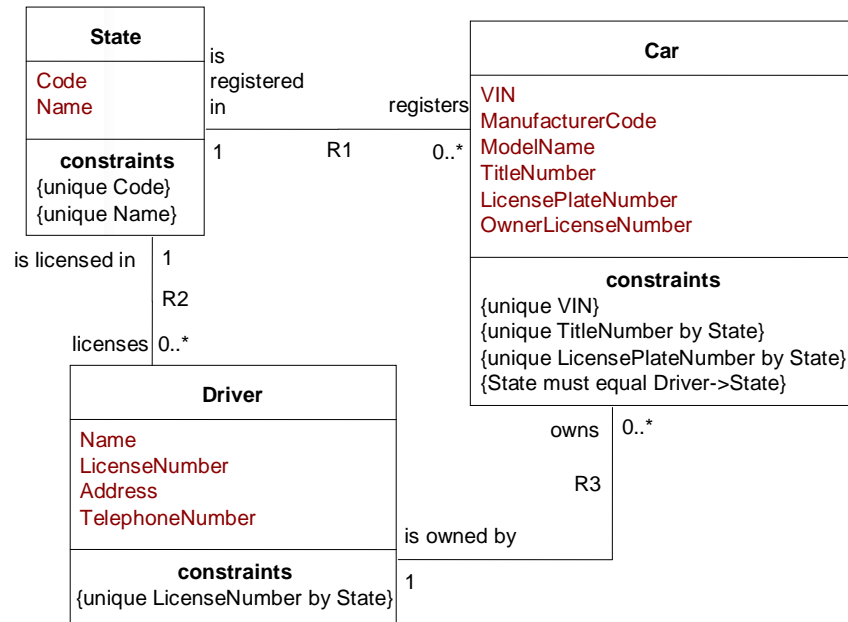
Precise information models must be capable of expressing and enforcing relationship constraints.

Consider the following slightly-contrived rule: An owner must register his car in the same state in which he is licensed to drive.”

With referential attributes, simply use one State referential attribute to formalize both the “registered to” and “owned by” relationships.



These kinds of problems that seem so well suited to referential attributes can also be expressed as constraints:



The full form of this constraint uses the verb phrases to make it easier to understand:

[is registered in] State **must equal** [is owned by] Driver → [is licensed in] State¹

Loop constraints in general can be expressed as either equal-set (as above) or subset constraints, as described in Section 8.4 of *Executable UML*. But that’s another paper.

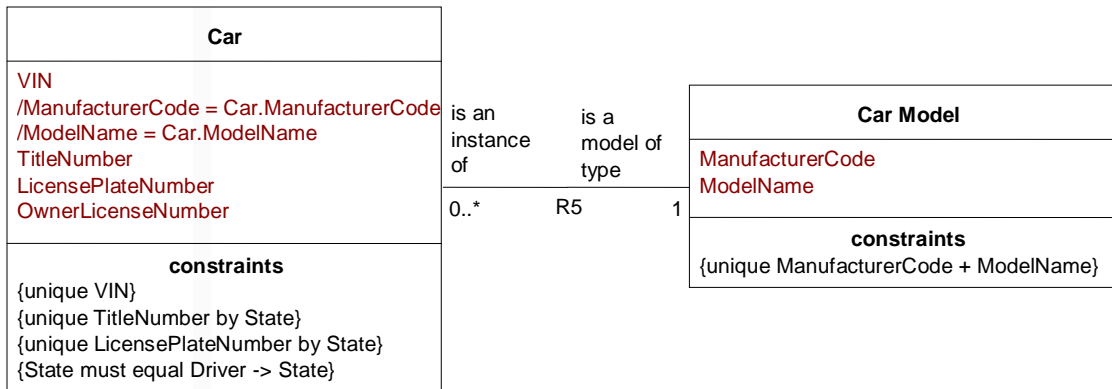
Referential Attributes are Derived Attributes

Although the rules of information modeling and the rules of relational data modeling (normalization) state that every attribute is supposed to be independent, there are cases in which an attribute is meaningful to a class even if its value can be completely determined from other attributes.

UML supports the notion of a “derived attribute” in which the attribute has a formula expression. *Executable UML* (3) has several examples of derived attributes, such as the total price of an Order.

If a modeler decides that it is important to include the identifier of a related class among the attributes of a class, such attributes should be defined as derived attributes. For example, the Manufacturer and Model Name of the Car, if they identify a Car Model, should be marked as derived (with the slash):

¹ This constraint expression makes it very clear that a related instance (or instances) is merely another characteristic of a class.



Metamodel

Lest you think that this constraint technique is just a fancy way of replacing formality with textual hokum, the constraint notation has a formal underlying representation.

In the metamodel an Identifier (uniqueness constraint) is composed of one or more IdentifyingProperties, each based upon a Property that is itself either an Attribute or an Association².

Similarly, a LoopConstraint is written as two LoopSideExpressions, each of which is a series of acyclic directed graphs with the same starting and ending classes.

Tooling Notes and Implications

It is vitally important that any tool that fully supports the method for purposes of model validation and execution record these individual constraints yet allow the constraints to be presented in either referential attribute or constraint expression notation. This will reinforce the notion that referential attributes, if shown, are really representations of the underlying constraints.

Summary

The referential attribute, a concept borrowed from relational data modeling's "foreign key constraint," has been a powerful technique for formalizing specific constraints in information models. However, referential attributes are not the only way to express these constraints. This paper has outlined a notation for expressing subpopulation identifiers and association constraints using a succinct but formally-based constraint notation that eliminates the perceived redundancy in referential attributes.

² The specification of a constraint in terms of properties is identical to the way in which primary key and foreign key constraints are specified in relational databases.

References

1. **Shlaer, Sally and Mellor, Stephen J.** *Object-Oriented Systems Analysis: Modeling the World in Data.* Englewood Cliffs : Yourdon Press, 1988.
2. —. *Object Lifecycles: Modeling the World in States.* Englewood Cliffs : Yourdon Press, 1992.
3. **Mellor, Stephen J. and Balcer, Marc J.** *Executable UML: A Foundation for Model-Driven Architecture.* Boston : Addison-Wesley, 2002.