26 March 2020

Paweł Filipek

MAIN MENU

DESIGN PATTERN / DOMAIN-DRIVEN DESIGN

Domain-Driven Design #06: aggregate and aggregate root

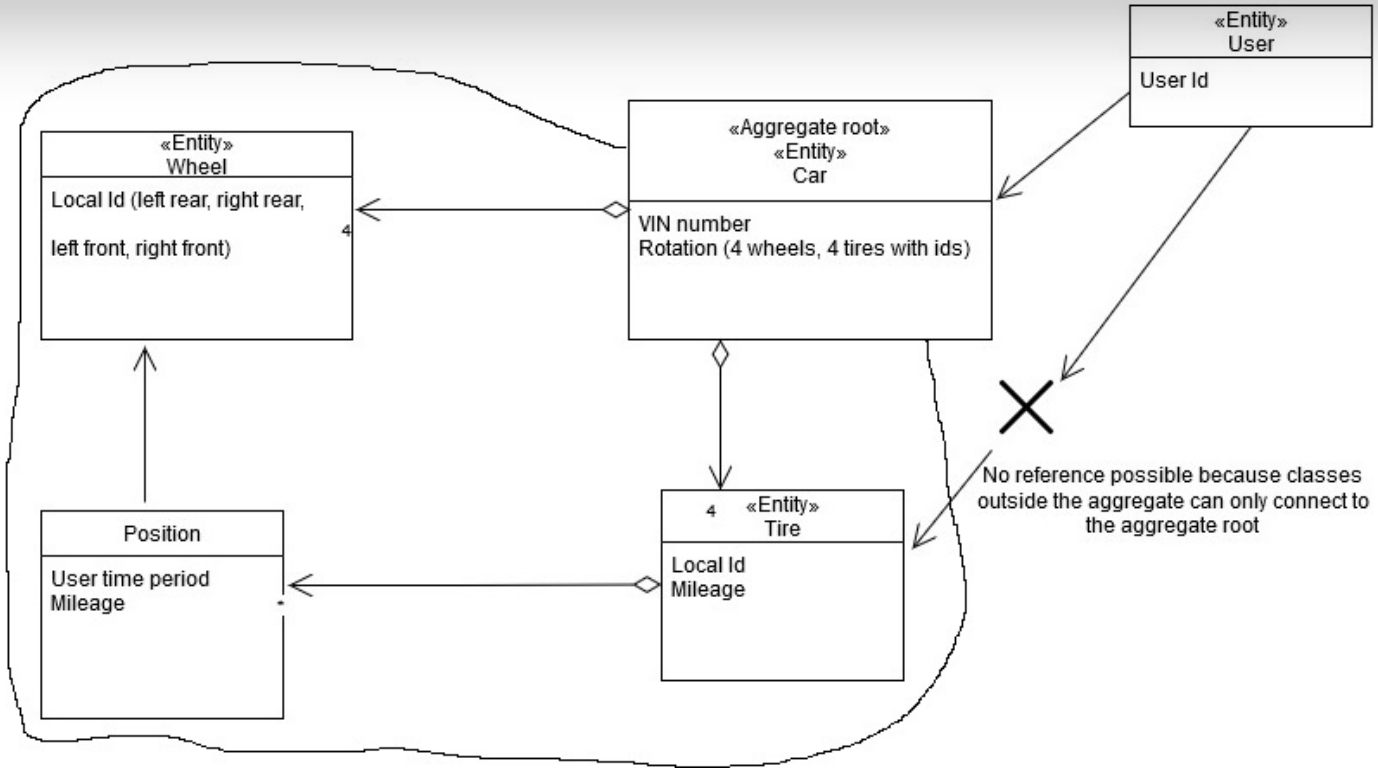
31 August 2019 - by **Paweł Filipek** - Leave a Comment

Designing a domain model with as few relationships as possible prevents the code from being complicated and difficult to maintain. On most large systems, relationships between entities cannot be avoided. In a situation when we remove a user from the database and he has references to the address, then we also have to delete. However, if another user also had a link to this address, incompatibility arises and we lose data. If we leave the address without deleting, it may happen that we will have a lot of redundant addresses that are not connected to any user. In this case, it is difficult to maintain consistency of references between objects in the domain model.

At first glance, this problem looks technical because it is related to the use transaction on the database side. This is really a domain model problem in which a certain pattern has no clearly defined boundaries. To remedy this, use abstract reference encapsulation in the model. For this purpose, the aggregate should be used, which is a set of related objects treated as one flowchart. Each aggregate has a root and boundary.

The root is a single specific entity contained in the aggregate. This object as the only one in the aggregate can have references to classes outside the aggregate limits. Local entities within the aggregate may have references to each other.

The car workshop model includes the global entity of the car. It is important that each car is a unique entity – the distinction is the VIN (vehicle indetification number) of the car. You need information about the mileage and tread wear for each tire in the car and history to which of the four wheels the tire was mounted. The tire as well as the wheel are entities, because the wheel has four positions and the tire must be recognized if it is old or new.



Summary

When creating an aggregate, we should observe the following principles:

– the aggregate root, i.e. the entity class, must have a global identity, thanks to which it is possible to attach object references from outside the aggregate;

– non-root entities have local identities;

TAGGED AGGREGAT AGGREGATE ROOT DDD DESIGN PATTERN DOMAIN DRIVEN DESIGN ERIC EVANS

RELATED POSTS

29 September 2019

Domain-Driven Design #04: entity

28 June 2019

Domain-Driven Design #05: value object

Domain-Driven Design #07: layers of architecture and persistence ignorance

About Paweł Filipek

View all posts by Paweł Filipek →

Leave a Reply

Your email address will not be published. Required fields are marked *

COMMENT

EMAIL *

WEBSITE

POST COMMENT