Username: Pralay Patoria **Book:** Under the Hood of .NET Memory Management. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Excessive References

Imagine your application has loaded into memory a complex domain model such as the one shown in Figure 4.8.

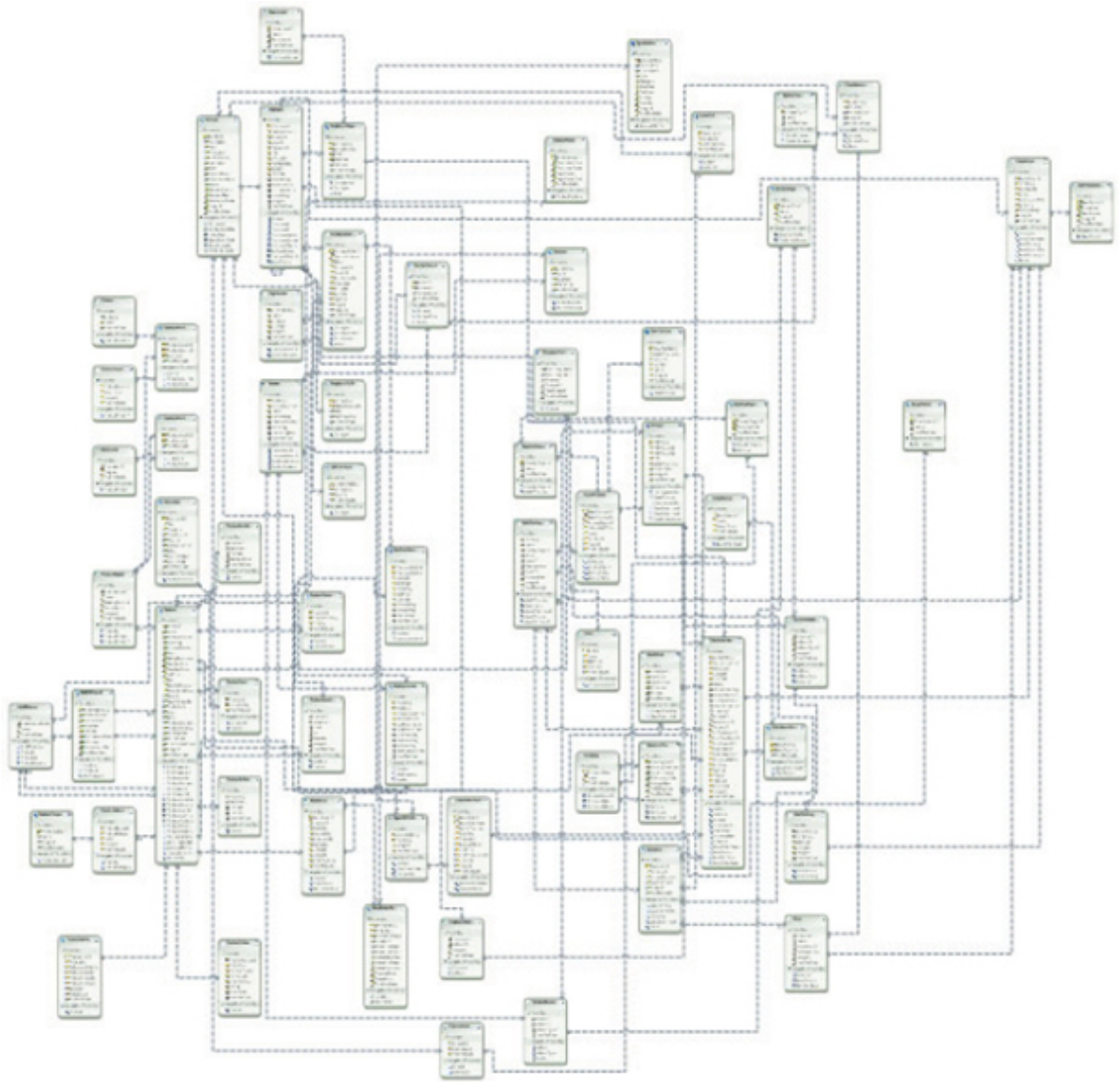

Figure 4.8: A complex domain model.

The GC must obviously analyze the entire graph to discover which objects can be collected. Complex models like that in Figure 4.8 will clearly tax the collector more than simple ones, which affects performance. The solution is to use architectural designs that load information as needed and release it when it is no longer necessary to maintain the reference.

Use `LazyLoading` to delay initializing related properties until they are referenced. For instance, if your domain model refers to a mortgage loan origination system, you may have domain objects related to the `Property` to `Borrowers` to `EmploymentHistory` to `CreditTransactions`, etc. All of which are relevant and necessary for much of your logic, but when you are calculating an amortization schedule, very little of that matters. If you load a loan object into memory to calculate the amortization schedule, you will need to pull out the loan amount, term, interest rate, and monthly payment - you *don't* need to load any details on the borrower or the property. Initializing only what is needed will reduce the complexity of your domain models and lessen the burden on the GC.