

## Re: limit of lines?

**Source:** <http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2004-12/7201.html>

---

**From:** Joanna Carter \ (TeamB) (joanna\_at\_nospam.co.uk)

**Date:** 12/30/04

Date: Thu, 30 Dec 2004 14:51:32 -0000

"Roman Kassebaum" <roman.kassebaumnospam@kdv-dt.de> a écrit dans le message de news: 41d3fe6f@newsgroups.borland.com...

- > *My framework (Now I'm careful, I do not call it OO) has several rules*
- > *considering the database:*
- > *1. For every entity (table in the db) there is a class (e.g. TPerson).*
- > *2. The class has for every attribute (field in the db) a published prop.*

And this is the fundamental mistake in your design. You should never derive an object from a database, this usually leads to very poor design. What is needed is to look at creating an Adapter mechanism which is essentially what an OPF (Object Persistence Framework) is. The class maps included in an OPF determine how the correct object design maps to the, usually less than perfect, database design.

- > *I use the person class in several programs which use the same entity*
- > *relationship model.*
- > *It is not possible to use the person class in other systems because the*
- > *person class has about 200 properties. We use it for a german pay role*
- > *system. Unfortunately german regulations are very complicated.*

Complicated or not, 200 properties has a 'bad smell' about it. You may well be putting properties in the Person class just because they are in the table, but this does not mean that, from an OO perspective, they belong there.

If you can't change the database schema to match the object design then you need to think about a class design that may require more complex mapping to cope with the database design; this may include mapping more than one class to a table.

- > *You are right. In this example the TOrder object would have a property*
- > *Customer: ICustomer and a property Lines: TOrderLineList.*
- > *Maybe it would be more elegant to manage this with a relation object.*

You only need to have a list property for the Lines in the Order class if that is what you mean.

borland.public.delphi.non-technical: Re: limit of lines?

- > *Sometimes I give the object a second property like CustomerId: TId*
- > *because the access to an Id is faster than the access to an Object or*
- > *Interface.*

If possible, you should avoid surfacing the ID in the business model as it has no relevance apart from as a means of identifying objects in the persistence mechanism. For this reason, I have developed a Value Type Framework in which all properties are backed, not by a simple field, but by a typed Value Type that allows me to get at the values behind the properties without having a public or published property. It is this ID that the database uses to keep itself in order.

- > *I do not load the Lines automatically. I will load them if someone calls*
- > *the lines property.*

I load the Line objects with at least just the ID to enable me to get the full objects quicker in the case of lazy loading.

- > *It is vice versa: I derive in the mammoth unit from so called custom*
- > *classes.*

I would seriously question the wisdom of that approach.

- > *That's why I talk to you, maybe a bit late.*

It's never too late :-))

- > *The framework handles referential integrity and the business logic. The*
- > *db is only a further security.*
- > *The framework has not special requirements to the db, that's why it*
- > *works with MSSQL, Oracle, Firebird and MySQL.*

If you consider the concept of an OPF, it should be possible to switch databases, even in mid-program. As long as the SQL generating code is hidden behind an abstract interface, there is nothing to say what storage you are really talking to; you might even be using text files :-)

- > *Forget include files. Derive a TPersonList from a TEntityList and*
- > *introduce the property Person: IPerson with the get and set method.*

How do you cope with Add, Remove, etc ? IF these are not typesafe as well, then this leaves the list as untypesafe because you have methods that don't require a definite type exposed; unless you only allow untyped access at the protected level of the base class.

- > *With include files you have problems with breakpoints, testing, Delphi*
- > *search in projects, include paths in make scripts.*

As I said, we have major size projects that use the Delphi template methodology and have had no serious problems.

Re: limit of lines?

borland.public.delphi.non-technical: Re: limit of lines?

> *Another point is that the exe size grows because the code is for every  
> include in the exe.*

Which is what happens for C++ where the template code is simply text-substituted at pre-compile; and I believe that C# will be similar. Think about it you have to have typesafe code symbols for every use of the template otherwise the compiler can't verify the types being used.

Having defended templates, there is nothing wrong with deriving from a properly protected base list class, as long as you don't mind the extra typing :-)

Joanna

--  
Joanna Carter (TeamB)  
Consultant Software Engineer  
TeamBUG support for UK-BUG  
TeamMM support for ModelMaker