# dotNetRDF Design Document

## dotNetRDF 0.4.3

Library: dotNetRDF.dll  
Version: 0.4.3  
Target Date: August 2011  
Author: Rob Vesse  
Proposed Implementer: Rob Vesse  
Last Updated:

## Required Features

* Reworked SQL backend support
* Virtualised Triples and Nodes

## Time Permitting Features

* None at present

## Known Issues/Bugs to Fix

* None at present

# Design

## Reworked SQL backend support

Firstly remove the defunct ISqlIOManager interface and all existing implementations and related classes.

Then design a new abstract base implementation that is a general ADO.Net implementation. Define the protected abstract implementation in terms of methods that get connections, commands, DataTable and DataReader objects.

Define a new Microsoft SQL Server implementation that is heavily based on stored procedures i.e. offload much of the heavy lifting of insertion onto the database server. Proposed procedures:

* CreateGraph
* DeleteGraph
* AssertTriple
* RetractTriple
* GetGraphTriples
* GetTriplesWith – all variants
* GetNode – gets the actual values needed to create a node based on an ID

Allow the implementation to operate in either materialised/virtual mode. In materialised mode retrieving Triples gets the full Nodes values for all Nodes. In virtual mode it takes advantage of the new virtualised storage implementation so that only IDs are retrieved and the values only get retrieved when needed.

Ensure that appropriate caching is used at the API end to avoid both unnecessary trips to the database and so that when operating in virtual mode a Node ID always returns the same virtual node instance (as far as possible).

## Virtualised Triples and Nodes

Define new INode implementations that are entirely virtual, they should be generic where the type parameter is the type of the ID that they use to refer to nodes and graphs in their underlying storage. These implementations **must not** inherit from the classes they are virtualising as they must have explicit casts available that let them be cast to the appropriate type. Once a cast has happened an implementation should cache the cast of itself internally so it can be reused for future casts.

Define a generic interface IVirtualisedRdfProvider (probably needs a better name?) which all the above must be instantiated with an instance of. The virtual INode implementations will then use this interface to load their actual values when required.