Thread-safety
As already discussed a few times, TDataFrame, TBufferMerger and Athena have their freedom reduced because of thread-safety issues in list of cleanups.
The upcoming read-write lock will make this safe.
Relevant jira issues: ROOT-8903 ROOT-8853 ROOT-8888

Performance
Adding and removing objects from the global lists has an overhead that becomes unsustainable when the number of objects and directories grows large.
In the case of a benchmark CMS analysis (which can be found at the link below) runtimes on a small part of the data were reduced from ~180s to ~1.5s by opting out of garbage collection.
We need to be able to opt out at TObject construction time or even before.
Relevant jira issue: ROOT-8910

Teachability, documentation
Memory management of garbage collected ROOT objects is hard to use correctly and easy to use incorrectly.
We need well documented guidelines of how to avoid performance issues, i.e. how to opt out of the garbage collection mechanism.
Ideally, these guidelines will be easy to follow correctly and hard to get wrong.
Utopically, there should be one single common opt-out mechanism.

Related: we have the responsibility to let users know what is safe and what is not - if Attila fell into the trap, anyone can:

- what does `EnableThreadSafety` actually allow users to do?
- is the "One file, one thread" golden rule documented anywhere?
- TChain *is a* TTree, but its memory management rules are different (TChains are in the list of specials)?

The current ownership model is described here (ROOT's user guide). This is, in my opinion, unteachable.

Appendix: stuff that I don't understand

A. Is this safe?

```
int main() {
 std::unique_ptr<TFile> f(TFile::Open("/tmp/f.root"));
 std::unique_ptr<TTree> t(static_cast<TTree*>(f->Get("t")));
 // does t->ResetBit(kMustCleanup); do anything?
 // does t->ResetBit(kCanDelete); do anything?
 return 0;
}
```

B. Is this safe?

```
int main() {
 std::thread t1([] { std::unique_ptr<TFile>
f(TFile::Open("/tmp/f.root")); });
 TH1F h("h","h",100,0.,1.);
 t1.join();
```

```
 return 0;
}
```