Garbage collectors

By Ann Harrison

It is no longer true that "every" transaction participates in garbage collection. In the olden days, before InterBase® 5, all garbage collection was cooperative. Each transaction looked at each record it read, and if it found unnecessary back versions, stopped whatever it was doing and removed them.

That behavior had the "unfair" effect of charging a transaction that did not change the database with lots of I/O spent cleaning up after transactions that did make changes. In V6, InterBase® introduce a "garbage collect thread" for Superserver only.

When the garbage collect thread is enabled, transactions identify unneeded back versions and put them on a list to be removed. When the system is idle, a special thread starts, reads the list, and starts cleaning up. The theory was that garbage collection would happen during slow times and not affect performance. Like many theories, this one has a flaw. Garbage collection is cheap if the back version to be removed is on the same page with the version of the record that is staying. There's only one page to change, and there are no tricky interactions with careful write. Normally, back versions are stored on the same page with the most recent record version. If that page fills up, then back versions need to go elsewhere, and the cost of storing and removing them increases enormously.

So, in a busy system, the garbage collect thread doesn't run often enough, back versions accumulate, and performance degrades markedly.

Vulcan disabled the garbage collect thread and performance is more even. Firebird 2 implements a hybrid mode for Superserver in which threads remove back versions themselves if the back version is on the same page with the primary record version. If not, the record goes on a list for the garbage collector. At some point, we'll test the various methods and pick the one that works best under load.