At the time of this writing, the KDE 4.1 release is imminent, and the Akonadi team is excitedly looking forward to the reaction of the many application and library developers in the wider KDE community who comprise its target audience. Interest in writing access resources for various storage backends is increasing, and people have started work on support for Facebook address books, Delicious bookmarks, the MS Exchange email and groupware server via the OpenChange library, RSS blog feeds, and others. It will be fascinating to see what the community will be able to create when data from all of these sources and many others will be available easily, pervasively, and reliably; when it will be efficiently queryable; when it will be possible to annotate the data, link items to each other, and create meaning and context among them; and when they can exploit that richness to make users do more with their software and enjoy using it more.

Two related ideas for optimization remain unimplemented so far. The first is to avoid storing the payload data in blobs in the database by keeping only a filesystem URL in the table and storing the data itself directly on the filesystem, as mentioned earlier. Building on that, it should be possible to avoid copying the data from the filesystem into memory, transferring it through a socket for delivery to the client (which is another process), thus creating a second in-memory copy of it only to release the first copy. This could be achieved by passing a file handle to the application, allowing it to memory map the file itself for access. Whether that can be done without violating the robustness, consistency, security, and API constraints that make the architecture work remains to be seen. An alternative for the first part is to make use of an extension to MySQL for blob streaming, which promises to retain most of the benefits of going through the relation database API while maintaining most of the performance of raw file system access.

Although the server and KDE client libraries will be released for the first time with KDE 4.1, the intent is still to share it with as much of the Free Software world as possible. To this end, a project on Freedesktop.org has been created, and the server will be moved there as soon as that process is finished. The DBUS interfaces have all been named in a desktop-neutral fashion; the only dependency of the server is the Qt library in version 4, which is part of the Linux Standard Base specification and available under the GNU GPL for Linux, Windows, OS X, and embedded systems, including Windows CE. A next major step would be to implement a second access library—in Python, for example, which comes with a lot of infrastructure that should make that possible with reasonable effort, or maybe using Java, where the same is true.

ThreadWeaver

ThreadWeaver is now one of the KDE 4 core libraries. It is discussed here because its genesis contrasts in many ways with that of the Akonadi project, and thus serves as an interesting comparison. ThreadWeaver schedules parallel operations. It was conceived at a time when it was technically pretty much impossible to implement it with the libraries used by KDE, namely Qt. The need for it was seen by a number of developers, but it took until the release of Qt 4