reasons should be handled under the hood as much as possible. The example of avoiding notification storms is taken care of by a configurable notification compression and update monitoring system, which users of the system can subscribe to with some granularity.

The next version of the high-level architecture diagram that was drawn, shown in Figure 12-2, thus reflects this notion by portraying the layers of the system as concentric rings or parts of rings.

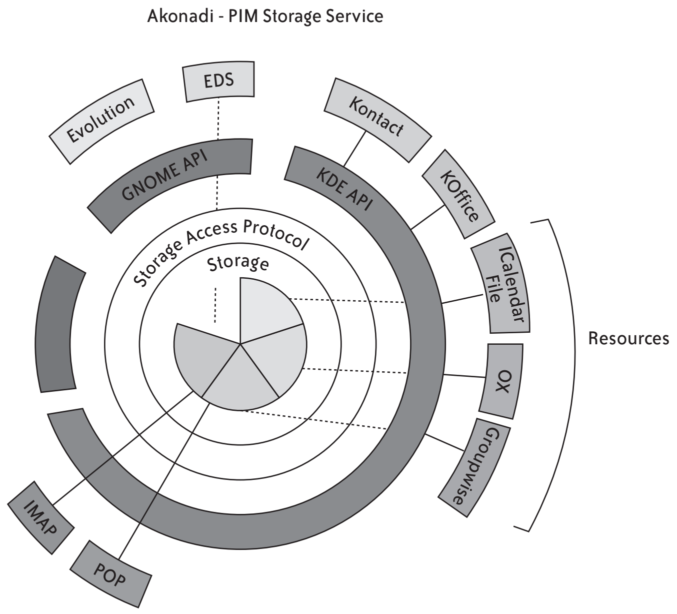

FIGURE 12-2. Inside out, not bottom up

Given the requirements outlined so far, it was fairly obvious that a relational database would make implementation of the lowest (or innermost) layer much easier. At least for the metadata around the actual PIM items, such as retrieval time, local tags, and per folder policies, just to name a few, which are typed, structured, and benefit from fast, indexed random access and efficient querying, no other solution was seriously considered. For the payload data itself, the email messages, contacts, and so on, and their on-disk storage, the decision was less clear cut. Since the store is supposed to be type-independent, a database table holding the data would not be able to make any assumptions about the structure of the data, thus basically forcing it to be stored as BLOB fields. When dealing with unstructured (from the point of view of the database) data, only some of the benefits of using a database can be leveraged. Efficient indexing is basically impossible, as that would require the contents of the data fields to be parsed. Consequently querying into such fields would not perform well. The expected access patterns also do not favor a database; a mechanism that handles continuous streaming of data