HTTP connections adds latency and overhead to the developer's own pages. His own database also offers a higher granularity of access than do the few dozen methods in the Facebook Platform API. Using his own data and a familiar query language such as SQL allows him to select only certain fields of a table, sort or limit results, match on alternative indices, or nest queries. If the platform's API does not offer the developer the ability to do intelligent processing on the platform's server, the developer must often import a superset of the relevant data and then do these standard logical transforms on his own servers after receiving it. This can be a significant burden.

PRODUCT PROBLEM: Obtaining data from the Facebook Platform APIs incurs much more cost than obtaining internal data.

As more traffic or usage starts to flow through an application consuming an outside data platform, factors such as bandwidth consumption, CPU load, and request latency can start to add up quickly. Surely this problem must have at least a partial solution. After all, haven't we optimized this in the data layer of our own single application stack? Isn't there a technology that enables fetching multiple sets of data in one call? How about doing selection, limiting, and sorting in the data layer itself?

DATA SOLUTION: Implement external data access patterns using the same one employed for internal data: a query service.

Facebook's solution is called FQL, detailed later in the section "FQL." FQL bears a great deal of resemblance to SQL, but casts platform data as fields and tables rather than simply loosely defined objects in our XML schema. This gives developers the ability to use standard data query semantics on Facebook's data, which is probably the same way they get to their own data. At the same time, the benefit of pushing computation to the platform side mirrors the benefits of pushing operations to the data layer in SQL. In both cases, the developer consciously avoids paying the price in his application logic.

FQL represents yet another improved data architecture based on Facebook's internal data, and is the next step after standard black-box web services. But first, we mention an easy and obvious way for a platform developer to eliminate the round-trip load of many data requests, and show why this is ultimately insufficient.

Method Call Batching

The simplest solution to load problems is something akin to Facebook's `batch.run` API. This eliminates the round-trip latency of multiple calls to *http://api.facebook.com* over the HTTP stack by accepting input for multiple methods in one batch, and returning the outputted XML trees in one response. On the client side, this flow translates to something like the code in Example 6-12.