WHITE PAPER

THQL

Overview

Sometimes institutions find it necessary to share data with outside entities. In clinical medicine, for example, observational studies may be derived from multiple hospitals' case-series: in the absence of controlled trials, pooling data from more than one hospital can permit the comparative effectiveness of different clinical interventions to be quantified (with some degree of precision, even if not at the level of double-blind trials). Or, in environmental science, ecological assessments from multiple jurisdictions may be merged into a holistic picture covering a large geographic area. Consider such data-integration challenges from a researcher's point of view: placing individual data-sources in context alongside parallel information from analogous locations makes each single data-source more valuable. It therefore behooves organizations (whether commercial or governmental entities) to acquire technologies which facilitate their participation in such data-sharing initiatives.

Multi-site data sharing projects can take many different forms, so for purposes of tighter focus this paper will concentrate on data-sharing architectures which have some common features. First, it is assumed that an organization seeks to provide restricted, selective access to certain internal data — this paper will not address public APIs or other technologies allowing wide-ranging, unmoderated general access to institutional data. Second, it is assumed that each organization's data-sharing protocols may be designed on a case-by-case basis. That is to say, data-sharing in the sense relevant to this paper does not involve a single, fixed technology, such as a web service, which upon being deployed provides access to some subset of organizational data according to a pre-determined, mostly unchanging protocol. Instead, this discussion will be addressed more to cases where organizations provide access to information under the aegis of particular projects, with specific goals and guidelines which shape the data-sharing engineering. Again, biomedical informatics offers a good example — suppose hospitals agree to share case series for Covid-19 patients, as part of a project to compile multi-site observational studies. How each hospital's clinical data is exposed for such a project, and the sort of information provided, should be determined by the focal topics of the observational analysis being conducted.

In short, preparing for data-sharing scenarios is not primarily, or exclusively, a matter of setting up a large-scale and immutable service which outside parties utilize in an open-ended manner; instead, organizations need the capability to build more focused portals which share data whose profile and organization is targeted to individual data-sharing projects. It may be the case that each such project requires its own software component, serving as a layer and adapter intermediary between the organization's internal data sources and outside parties' applications which request data in the context of a given multi-site project.

So as to analyze correct design-patterns for such adapter-components, the current discussion will focus on solutions that can be treated as extensions to database engines. This approach is somewhat abstracted from concrete implementations, because it is not necessarily the case that all data from a given organization would be housed in one single database. Instead, large institutions can easily have decentralized, heterogeneous information spaces spanning multiple forms of database design, as well as assets maintained outside any database at all (e.g., via filesystems or network-connected software). However, protocols for data-sharing projects meeting the criteria relevant for this paper may be formalized in the context of accession to individual database instances, and subsequently generalized to more complex information-storage scenarios.

Assume, accordingly, that the goal of selective, curated outside-party access to some portion of an information space is manifest specifically in the context of a single database instance, implemented via a specific database engine. The question is then how to architect or enhance the database engine so that programmers may implement selective views onto the database instance, permitting remote access to relevant data whose extent and structure is determined by the parameters of a given data-sharing project.

As a framework for analyzing this problem-space, the current paper will concentrate on one specific kind of database architecture, specifically that of "hypergraph" database engines. Hypergraph databases are well-suited to multi-faceted, heterogeneous data profiles which combine features of more restricted database designs. As a result, protocols developed in the hypergraph context may be adapted to apply to other genres of database as well (techniques for doing so will be discussed below). In sum, this paper will examine the problem of formulating a data-sharing protocol in the specific context of hypergraph database engines, an then consider how such a protocol may be extended to other data-persistence environments.

Data-Sharing Protocols in the Hypergraph Database Context

The central theme of paper, then, is as follows: we have a hypergraph database, and intend to make some part of the data which it contains available within the context of targeted data-sharing projects. Hypergraph database engines are chosen because they support heterogeneous, flexible data profiles — information need not be constrained to a single format — and because application-integration is often easier via the hypergraph architecture than via alternatives. That is, less development time or biolerplate code is needed to marshal data between application-runtime and database-persistent representations. Insofar as these are benefits of hypergraph database engines themselves, they also represent features which are reflected in protocols for outside database access. That is to say, how external applications obtain data remotely from hypergraph database instances should reflect patterns in how hypergraph data is organized.

At the same time, there are several different hypergraph database engines used in production, each with distinct features and paradigms. As a result, we cannot point to one single, canonical hypergraph database model that could provide a common protocol applicable to all relevant technologies. Instead, a multi-purpose protocol should integrate features specific to different individual engines. The goal is, therefore, to formulate a system for describing remote hypergraph database access which could be applied as a mold for software components obtaining data from a remote portal backed by one of several different hypergraph engines. The same accession framework should apply to multiple engines, integrating each one's distinct modeling features.