Data Management Support for Notification Services

Wolfgang Lehner

Technische Universität Dresden, Germany
lehner@inf.tu-dresden.de

Abstract. Database management systems are highly specialized to efficiently organize and process huge amounts of data in a transactional manner. During the last years, however, database management systems have been evolving as a central hub for the integration of mostly heterogeneous and autonomous data sources to provide homogenized data access. The next step in pushing database technology forward to play the role of an information marketplace is to actively notify registered users about incoming messages or changes in the underlying data set. Therefore, notification services may be seen as a generic term for subscription systems or, more general, data stream systems which both enable processing of standing queries over transient data. This article gives a comprehensive introduction into the context of notification services by outlining their differences to the classical query/response-based communication pattern, it illustrates potential application areas, and it discusses requirements addressing the underlying data management support. In more depth, this article describes the core concepts of the *PubScribe* project thereby choosing three different perspectives. From a first perspective, the subscription process and its mapping onto the primitive publish/subscribe communication pattern is explained. The second part focuses on a hybrid subscription data model by describing the basic constructs from a structural as well as an operational point of view. Finally, the *PubScribe* notification service project is characterized by a storage and processing model based on relational database technology.
To summarize, this contribution introduces the idea of notification services from an application point of view by inverting the database approach and dealing with persistent queries and transient data. Moreover, the article provides an insight into database technology, which must be exploited and adopted to provide a solid base for a scalable notification infrastructure, using the *PubScribe* project as an example.

1 Introduction

The technological development in recent years has created an infrastructure which enables us to gather and store almost everything that can be recorded. However, the stored

T. Härder and W. Lehner (Eds.): Data Management (Wedekind Festschrift), LNCS 3551, pp. 111-136, 2005.

data gets frequently lost in the existing varieties of databases. The main reason is that nobody is willing to browse multiple databases and specifically search these databases for certain entries.

Multiple methods from a database-technological point of view are trying to leverage this very general problem. The context of information integration [26] tries to come up with an either logically (multi-database systems or virtual database systems) or physically integrated data set, often seen in data warehouse environments [16]. From a more application-oriented point of view, methods and techniques coming from the area of knowledge discovery in databases try to generate hypotheses which might be of some interest for the users of the data set. The real benefit of this approach is that a user may specifically focus on these results as a starting point for an interactive analysis [10, 19, 9].

A completely different approach is taken when inverting the current way of interacting with databases by moving from a *system-centric* to a *data-centric* behavior [22, 23]. The query-based approach follows the request/response paradigm, where the user (or client) is posing a query and the (database) system tries to execute the query as fast as possible. The result is delivered via basic interfaces (like ODBC, JDBC, or CLI) to the user's application context. Fig. 1a illustrates this interaction pattern with database management systems on the one side acting as data providers and clients on the other side acting as data consumers. Both parties are in a close relationship with each another, i.e. every client has to know the location and the context of the specific database.

1.1 Publish/Subscribe as the Base for Notification Systems

Inverting the "request/response" idea leads to the communication pattern very well known as "publish/subscribe" [5], which is used in various situations. For example, publish/subscribe is utilized in software engineering as a pattern [12] to connect individual components. In the same vein, publish/subscribe may be used to build a data-centric notification service. Notification services consist of publishers, subscribers, and finally, a (logically single) notification brokering system. Fig. 1b gives a conceptual overview of the scenario. On the one side, publishing systems (or publishers) are acting as data providers, generating information and sending data notifications to the brokering component as soon as the information is available. Notifications are collected, transformed into a pre-defined or pre-registered schema and merged into a global database. Depending on the type of subscriptions (section 3.1), notifications may remain in the database only for the time span required to notify interested subscribers.

On the other side, subscribers—acting as data consumers—are registering "interest" (information template, profile, ...) providing the delivery of certain notifications using specific formats and protocols with a pre-defined frequency. The notion of interest is manifold and will be further discussed in the following section. Furthermore, query specification and result transmission are decoupled from a subscriber's point of view. Once a query is formulated, the user is no longer in contact with the notification system but receives a notification only if new messages of interest have arrived at the database system or if a given time interval has passed. The advantage for the user is tremendous: