Reweaving the Tapestry: Integrating Database and Messaging Systems in the Wake of New Middleware Technologies

Sangeeta Doraiswamy[1], Mehmet Altinel[1], Lakshmikant Shrinivas[2], Stewart Palmer[3], Francis Parr[3], Berthold Reinwald[1], C. Mohan[1]

1. IBM Almaden Research Center, San Jose, CA
2. University of Wisconsin-Madison, Madison, WI
(while on assignment at IBM Almaden Research Center)
3. IBM T.J. Watson Research Center, Hawthorne, NY
{dorais, maltinel, slp, fnparr, reinwald, mohan}@us.ibm.com, lshrinivas@wisc.edu

Abstract. Modern business applications involve a lot of distributed data processing and inter-site communication, for which they rely on middleware products. These products provide the data access and communication framework for the business applications.
Integrated messaging seeks to integrate messaging operations into the database, so as to provide a single API for data processing and messaging. Client applications will be much easier to write, because all the logic of sending and receiving messages is within the database. System configuration, application deployment, and message warehousing are simplified, because we don't have to manage and fine-tune multiple products.
Integrating messaging into a database also provides features like backup, restore, transactionality & recoverability to messages. In this paper, we'll look at some aspects of messaging systems, and the challenges involved in integrating messaging such as message delivery semantics, transaction management and impact on query processing.

1 Introduction

Over the last decade, we have been witnessing a sea change in computing platforms. Proliferation of Internet technologies, advances in consumer hardware products, and availability of pervasive network bandwidth have fueled the development of a new class of Web-based applications. The foundation of this revolution was laid down by the dramatic advances in middleware technologies and products. New middleware standards such as J2EE [7], and .NET [9], and service-oriented architectures like Web Services [16] have stimulated the development of highly dynamic e-Business infrastructures. Among the many services and components in this new framework, messaging systems

T. Härder and W. Lehner (Eds.): Data Management (Wedekind Festschrift), LNCS 3551, pp. 91-109, 2005.

have emerged as a key component of this enterprise infrastructure. The current trend in middleware architectures indicates that messaging systems will strengthen their role in future middleware infrastructures towards so-called Message-Oriented Middleware (MOM) systems.

Early messaging systems functioned solely as queue managers. In this limited role, they offered queued transaction processing and provided basic services such as security, configuration, performance monitoring, recovery, etc. The common belief was that database systems already covered these features and a full-function messaging system only duplicates them [6]. This viewpoint changed dramatically as messaging systems evolved in the midst of middleware infrastructures. Message queuing systems focus on providing high performance transactional access in cases where there is a high rate of message insertion and deletion to a large number of independent destinations where on average there are few or even zero messages waiting to be received on any individual destination at any point in time. Their new features are sufficiently more complex and sophisticated that providing the same level of functionality in a database system is out of the question. Extended delivery models, highly scalable asynchronous message handling, distributed network protocols and processing, message-specific multiple qualities of service are just a few features of advanced messaging systems. In general, it is not feasible to contain these typical messaging system features inside a database system. In its new role, a messaging system is a core member of an advanced middleware infrastructure. It is the backbone of middleware that glues all other middleware components together. New middleware architectures and standards clearly separate database systems as external data sources. In this model, messaging systems are decoupled from database systems and they utilize a database system only as a (relational) “data source”.

Messaging applications have the important characteristic that at any point in time, a significant amount of system and application data is in flight (i.e., stored) in message queues. Hence, messaging systems face two big data management challenges: (1) Coping with increased number of messages and sheer volume of data, (2) Processing the data in message queues to provide a real-time snapshot of the state of a business for critical decision making. Particularly, messaging systems are expected to provide elaborate, scalable data management functions such as powerful message selection (filtering), aggregation, transformation, etc., besides their traditional networking functionality. In other words, messaging systems have been slowly transformed into a new type of database system for transient and distributed data.

On the other side, database system vendors acknowledged the rise of middleware systems and realized the importance of seamless integration into middleware infrastructure. Several major database system vendors made the strategic decision to extend their database engines with core messaging capabilities. This way, they were better connected to a new class of applications, while applying scalable data management techniques (which they excelled in for the last 25 years) to messaging operations. Furthermore, they took advantage of built-in messaging support to develop more effective database services such as data replication or query notification services.

Despite the fact that early messaging systems had a broad range of common characteristics with database systems, the two systems developed and grew independently as stand-alone systems. Messaging systems are characterized by different inherent se-