

Mobile and Embedded Databases

Anil Nori

Microsoft Corporation

One Microsoft Way, Redmond, WA 98052-6399

anilnori@microsoft.com

ABSTRACT

Recent advances in device technology and connectivity have paved the way for next generation applications that are data-driven, whose data can reside anywhere, can be accessed at any time, from any client. Also, advances in memory technology are driving the capacities of RAM and Flash higher, and their costs down. These trends lead to applications that are mobile, embedded, and data-centric.

This tutorial presents an overview of the mobile and embedded database systems and their applications.

Categories and Subject Descriptors:

H.2 [Database Management], H.3 [Information Storage and Retrieval], H.4 [Information Systems Applications]

General Terms: Design, Management

Keywords: Devices, Embedded, Extensible, In-Memory database, Streams, Mobile, Synchronization

1. THE NEW ENVIRONMENT – MOBILE, DISCONNECTED, AND EMBEDDED

Recent advances in processors, memory, storage, and connectivity have paved the way for next generation applications that are data-driven, whose data can reside anywhere (i.e. on the server, desktop, devices, embedded in applications) and that support access from anywhere (i.e. local, remote, over the network, from PDAs, in connected and disconnected fashion). Memory sizes have gone up and prices have come down significantly; with 64 bit addressability, it is not uncommon to configure servers with 8 – 16GB of memory, and desktops with 2 – 4 GBs of memory. With advances in Flash memory technology, large flash drives are available at reasonable prices. Computers with 32 GB flash drives are making way into the market. Flash drives not only eliminate seek time and rotational latency they consume significantly less power than the conventional disk drives, making them ideal for portable devices. All this naturally leads to disconnected, composite (composing other data and processes), and mobile application architectures. Application components can run on different tiers, in different service boundaries, and on different platforms (e.g. server, desktop, PDAs).

These new breeds of applications fall into one or more of the following categories:

Copyright is held by the author/owner(s).
SIGMOD'07, June 12–14, 2007, Beijing, China.
ACM 978-1-59593-686-8/07/0006.

1.1 Mobile

As more users adopt Wi-Fi enabled laptops, increasingly capable PDAs, and Smartphones, the need for mobile applications is increasing. Applications like Email, Calendaring, CRM (Customer Relationship Management) are already targeting mobile devices. Middleware infrastructures like application servers and workflow services are becoming mobile-aware. Some reasons for such mobility trends are:

- More employees are mobile
- Mobile usage is broadening. Mobile usage is already prevalent in certain vertical domains like Healthcare, Insurance, and Field Services.
- Email and offline access is becoming pervasive
- Mobile applications are more than just browser windows – more and more applications now run natively on mobile devices.

Data management and data access on mobile devices is central to mobile applications. As mobile applications achieve widespread adoption in the enterprise, mobile and embedded DBMSs – which are needed to support such applications – become an important part of the IT infrastructure. As these applications grow more disconnected and sophisticated, data sizes increase and the need for rich data processing capabilities increases.

1.2 Streaming

In certain scenarios, data is simply streamed intelligently through application logic. There is no additional need to store the data – except transiently, for the duration of the operation. Conventional databases however require the data to be first loaded into the database; the operation is then performed, and the data may be later removed from the database. All of this adds significant complexity to the application, and dramatically reduces its performance and throughput.

Consider, for example, RFID data in applications. RFID tags are portable sensors that communicate over specialized protocols with RFID reader devices. An increasing number of applications have begun to utilize RFID technology including Manufacturing (e.g. Vendor Managed Inventory, Assembly scheduling), Distribution (e.g. Goods delivery, Shipping verification), Retail (e.g. Shelf stocking, Receiving, Store Replenishment and Ordering, Theft, Merchandise Location), Healthcare Industry to identify patients, Tracking books and automated checkout in libraries, etc. In all these applications, RFID readers (devices) generate events when they identify RFID tags. The RFID event streams are filtered, aggregated, transformed, and correlated so

that the events can be monitored in real-time. For example, when a truck carrying packages of the product parts (with RFID tags) enters (or leaves) a warehouse, the readers generate events. Spurious events are filtered; related products are aggregated; the event data is appropriately transformed and presented on a monitoring dashboard. The event processing is data-centric and typically requires an in-memory rule engine and query processing for real-time processing.

1.3 Disconnected

Distributed and disconnected application architectures fundamentally change the way applications access and manage data. Instead of locking data in a central location, data is moved closer to the applications, and this enables data processing to be performed in an autonomous and efficient fashion. These applications may run in the mid-tier, on desktops, or on mobile devices. Such an environment is inherently disconnected – there is no need for continuous connectivity between the data sources. Data may be accessed from its original sources, transformed and cached (or stored) close to the applications.

For example, consider a product catalog application aggregating product information across multiple backend application and data sources. Operations against such an aggregate catalog require the operations to be decomposed into operations on the underlying sources; the underlying operations are invoked, the responses are collected, and the results are aggregated into cohesive responses to the end users and businesses. A typical Catalog Browse operation iterates over a large amount of product data, filters it, personalizes it, and then presents the selected data to the users.

These operations are data-centric i.e. they require accessing and querying backend data. Accessing large sets of backend data for every catalog operation can be prohibitively expensive, and can significantly impact the response time and throughput of the application. Caching and querying is therefore a key mechanism in application servers both for performance and functionality purposes.

1.4 Embedded

Many applications perform simple to moderate manipulation of data. They need a way of storing, retrieving and manipulating the data within the application. These applications themselves are not complex in nature, and are designed to meet a specific user need. Typically, these applications are developed by application vendors specializing in industry verticals (domains) – e.g. Healthcare, Finance etc. These vendors are domain experts but not database specialists. They would rather not spend their time in database installation, deployment, and management. Their focus is the application and everything is embedded within the application. While synchronization with backend databases is important, this is typically never seen by the application developer. The databases are typically single application databases; the applications do not want to share (or coordinate) their database with others (which is the case with general purpose database services). These applications could run on devices, tablets, desktops, or servers. The ideal way of deployment is deploying the database components along with the application as a single install. Additionally, different applications may have varying needs from the database – some require only ISAM access, while others require general query

support; still others may require synchronization with backend databases. Thus, componentization of database functionality is a key requirement of embedded applications.

While the vertical (embedded) application domain is rapidly growing, traditional DBMS vendors have not paid attention to their needs. Therefore, application vendors have used home grown components for data management or – worse - each developer used whatever is the technology of the day (s)he is familiar with. Files, XML, a rudimentary store and retrieve mechanism, or a dedicated team for creating an internal product for data management are some of the techniques employed to fill the need for an embedded database. Consequently, application developers are looking to DBMS vendors for better support for embedded databases.

2. NEED FOR SMALL, EMBEDDED, IN-MEMORY, MOBILE DBMS

The data access and data management requirements of the applications described above are significantly different from what traditional server DBMSs support. These new applications must be able to run on multiple tiers ranging from devices to servers to web. The next generation applications would benefit from various database mechanisms. However, database mechanisms (like query, indexing, persistence) must be unlocked from the traditional, monolithic DBMSs and made available as embeddable components (e.g. DLLs). These components can then be embedded within applications, thereby, enabling them to meet the requirements described above. We call such mobile and hostable database technology as “Mobile and Embedded DBMS Technology.” Mobile and Embedded DBMSs have the following characteristics:

1. **Embeddable in applications** – Such DBMSs form an integral part of the application or the application infrastructure, often requiring no administration. Database technology is delivered as part of the application (or app infrastructure). While the database must be embeddable as a DLL in applications, it must also be possible to deploy it as a stand-alone DBMS with support for multiple transactions and applications.
2. **Small footprint** – For many applications, especially those that are downloadable, it is important to minimize DBMS footprint. Since the database system is part of the application, the size of the mobile and embedded database system software affects the overall application footprint.
3. **Run on Mobile Devices** – The DBMSs that run on mobile devices tend to be specialized versions of mobile and embedded DBMSs. In addition to handling the memory, disk and processor limitations of these devices, these DBMS must also run on specialized operating systems. The mobile DBMSs must be able to store and forward data to the back-end DBMSs – synchronization with backend systems is critical for mobile applications.
4. **Componentized DBMS** – Often, to support the small footprint requirement, it is important to include only the (database) functionality that is required by the applications. For example, many simple applications just require ISAM like record-oriented access. For these applications, there is

no need to include the query processor, thereby increasing the footprint. Similarly, many mobile and mid-tier applications require only a small set of relational operators; some applications require XML access and not relational access. So, it should be possible to pick and choose desired components.

5. ***In-Memory DBMS*** – These are specialized DBMSs serving applications that require high performance on data that is small enough to be contained in main memory. In-memory DBMSs require specialized query processing and indexing techniques that are optimized for main memory usage. Such DBMSs also can support data that may never get persisted.
6. ***Portable databases*** – There are many applications which require very simple deployment – installing the application should install the database associated with it. This requires the database to be highly portable. Typically, a simple single file database (e.g. like Microsoft Access databases) is ideally suited for this purpose. There is no need to install the DBMS executable separately – installing the application installs the DBMS (e.g. it's a DLL) and then copying the database file completes the application migration.
7. ***Synchronize with back-end data sources*** – In the case of mobile data and cached data, it must be possible to synchronize the data with the back-end data. In typical mid-tier (application server) caches, the data is fetched from the back-end databases into the cache, operated on, and synchronized with the back-end database.

3. TUTORIAL OUTLINE

1. Introduction and scope of the tutorial
2. Platform environment
 - a. Device environment
 - b. Embedded systems
 - c. Large memory systems
 - d. Flash memory environment
3. Next generation Applications: Composite, distributed, mobile, and embedded
4. Application Architectures
 - a. Role of the databases
 - b. Need for embedded databases
5. Mobile and embedded database system architectures
 - a. Component architecture
 - b. Storage engine mechanisms
 - c. QP mechanisms
 - d. Extensibility
 - e. Synchronization

- f. Comparison with traditional (e.g. enterprise) database systems
6. Mobile and Embedded DB case studies: Features and Architectures
7. Use of Embedded databases
 - a. Device Apps
 - b. Desktop Apps
 - c. Large scale distributed caches
 - d. Stream Processing Systems
8. Research Opportunities – Challenges
 - a. Large Memories, Flash drives, multi-core processors
 - b. Real-time data processing
 - c. Compression
 - d. Etc.
9. Q/A, Discussion

4. DURATION

1.5 hours.

5. TARGET AUDIENCE

This tutorial is very practical and systems oriented. The tutorial is intended for database/applications researchers and practitioners who want to gain a comprehensive understanding of mobile, embedded, large memory systems, including hardware trends, applications, and database systems.

6. RELEVANCE TO AUDIENCE

We believe embedded databases will become a critical and integral part of the next generation applications, which are distributed, composite, and mobile. It is important for database researchers and practitioners to understand the current state of the art and the potential opportunities for further research. We believe continuing advances in the processing, memory, and storage trends present a great opportunity for innovation in this space.

7. PRESENTER BIO

Anil Nori is a Distinguished Engineer at Microsoft. He is an architect in the SQL Server organization, focusing on data and application platform technologies. His current areas of focus include: Mobile & Embedded databases, Caching/in-memory databases, Rich client/mid-tier data and application architectures, Data/Application hosting, and Data/App manageability. He is part of the SQL Server leadership overseeing the overall vision, strategy, and architecture for Microsoft data platforms. He has over 25 years of experience in building complex database and application systems.