A Step To Embedded Database

A Techno Change

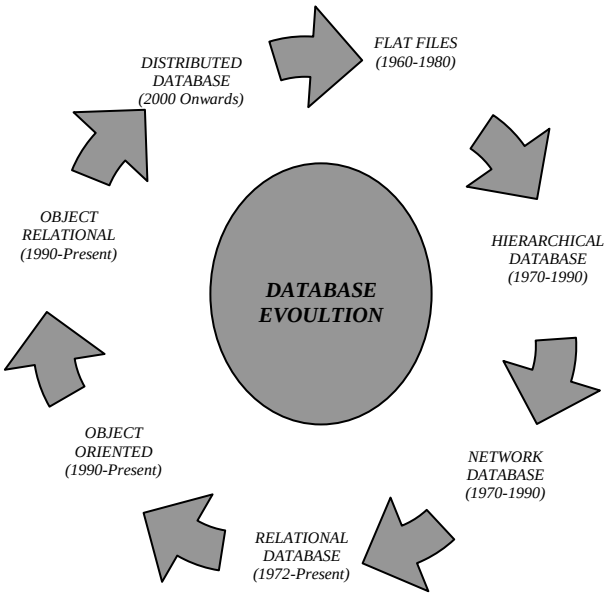
Manik Sharma
Assistant Professor & Head, PG Deptt. of computer Science and Applications
Sewa Devi SD College Tarn Taran
Manik_sharma25@yahoo.com

Abstract— Recent advances in device tools and connectivity have tiled the way for next generation applications that are data-driven, where data can reside anywhere, can be accessed at any time, from any client. Embedded systems are computers (microprocessors) that are enclosed (embedded) in customized hardware. An **embedded database** system is a database management system which is closely coupled with application software that requires access to stored statistics or data, such that the database system is "hidden" from the application's end-user and requires little or no ongoing maintenance. More than 20 years one could argue that since the beginning of software, embedded databases have been in existence. The operations of the embedded database are invoked by the application. The embedded database is embedded within an application either as in-line code or linked libraries unlike the traditional general purpose enterprise relational databases such as Oracle, DB2, and SQL Server etc which normally run as the separate applications that are independent of the system application. The key operational advantage of the embedded database is that using the embedded database the users and administrators are not burdened with time-consuming installations or maintenance as the database is packaged with the application and is generally self maintaining. The embedded databases can be relational, hierarchical, network model, XML based, object oriented etc. The embedded DBMS are typically used in the mobile phones, PDA's, set-top boxes, automotives etc.

Keywords- *Embedded Database, DBMS, Software, Oracle, mobile phones*

1. INTRODUCTION

As we know that the database is defined as collection of interrelated data. Initially small databases were first developed or funded by the U.S. government for agency or professional use. In the 1960s, some databases became commercially available, but their use was funneled through a few so-called research centers that collected information inquiries and handled them in batches [1]. Recent advances in device technology and connectivity have paved the way for next generation applications that are data-driven, where data can reside anywhere, can be accessed at any time, from any client. Embedded systems are computers (microprocessors) that are enclosed (embedded) in customized hardware. Examples of embedded systems are portable medical equipment, cellular phones, or consumer electronics items. An **embedded database** system is a database management system which is closely tied with application software that requires access to stored data, such that the database system is "hidden" from the application's end-user and requires little or no ongoing maintenance [2]. More than 20 years one could argue that since the beginning of software, embedded databases have been in existence.



In other words we can say that embedded database is embedded in some another software applications. The operations of the embedded database are invoked by the application. The embedded database is embedded within an application either as in-line code or linked libraries unlike the traditional general purpose enterprise relational databases such as Oracle, DB2, and SQL Server etc which normally run as the separate

applications that are independent of the system application.

The key operational advantage of the embedded database is that using the embedded database the users and administrators are not burdened with time-consuming installations or maintenance as the database is packaged with the application and is generally self maintaining. The embedded databases can be relational, hierarchical, network model, XML based, object oriented etc.

The embedded DBMS are typically used in the mobile phones, PDA's, set-top boxes, automotives etc.

Worth of Embedded Database: Modern embedded devices are now responsible for storing more data than ever before. Some devices get an edge on the competition by synchronizing data without interrupting normal use. Important data must not be lost to corruption caused by a power failure. For these devices, performance and reliability are critical.

2. EMBEDDED DBMS CHARACTERISTICS

The data access and management requirements of the applications described above are significantly different from that of traditional server DBMS. These new applications must be able to run on multiple tiers ranging from devices to servers to web and would benefit from various existing database mechanisms. However, these database mechanisms (like query, indexing, persistence) must be unlocked from the traditional monolithic DBMS and made available as embeddable components (e.g. DLLs) that can be embedded within applications, thereby, enabling them to meet the requirements described above. Such Mobile and Embedded DBMS have the following characteristics:

1. **Embeddable in applications** – Mobile and Embedded DBMS form an integral part of the application or the application infrastructure, often requiring no administration. Database functionality 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 DBMS affects the overall application footprint. In addition to the small footprint, it is also desirable to have short code paths for efficient application execution. Most of these applications do not require the full functionality of commercial DBMSs; they require simple query and execute in constrained environments.

3. **Run on mobile devices** – The DBMS that run on mobile devices tend to be specialized versions of mobile and embedded DBMS. In addition to handling the memory, disk and processor limitations of these devices, the DBMS must also run on specialized operating systems. The DBMS must be able to store and forward data to the back-end databases as synchronization with backend systems is critical for them.

4. **Componentized DBMS** – Often, to support the small footprint requirement, it is important to include only the 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 while others require XML access and not relational access. So, it should be possible to pick and choose the desired components.

5. **Automatic DBMS** – The embedded DBMS is invisible to the application user. There can be no DBA to manage the database and operations like backups, recovery, indexing, tuning etc. cannot be initiated by a DBA. If the database crashes, the recovery must start instantaneously. The database must be self managed or managed by the application. Also, embedded DBMS must auto install with the application it should not be installed explicitly (user action) or independently. Similarly when the application is shutdown, the DBMS must transparently shutdown.

6. **In-Memory DBMS** – These are specialized DBMS serving applications that require high performance on data that is small enough to be contained in main memory. In-memory DBMS require specialized query processing and indexing techniques that are optimized for main memory usage. Such DBMS also can support data that may never get persisted.

7. **Codeless database** – Portable database should be free from any threat. The executable code can become the reason for malfunctioning or destruction of data in the form virus. By eliminating any code storage in the database, we can make our database consistent and safe [4].

8. **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, single file databases (e.g. like Microsoft Access databases) are ideally suited for this purpose. Again, there should be no need to install the DBMS separately – installing the application installs the DBMS and then copying the database file completes the application migration. With the help of portable database [5] application we can reduce the anomalies in database migration.

9. **Synchronize with back-end data sources** – In the case of mobile and cached scenarios, it must be possible to synchronize the data with the back-end data sources. 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.

10. **Remote management** – While mobile and embedded DBMS must be self managed, it is important to allow them to be managed remotely also, especially those on mobile devices. In enterprises (e.g. FedEX, UPS), mobile devices must be configured and managed in a manner compliant with the company

3. Embedded software in India

Typically software for embedded systems need to have a very small footprint (i.e. be able to run in a small amount of memory) and often have to work in real-time. Companies here in India offer specialized operating systems and languages, which make this possible. These companies ensure the designing, developing, and testing of software for embedded systems and components meet specific customer requirements. They use diverse, real-time operating systems, devices and platforms and associated embedded tools and technologies. The various Embedded Software domains[5] in India are:

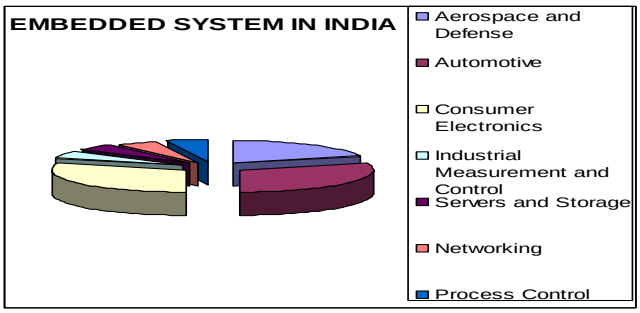


Figure2: Embedded System in India

The various embedded database system in commercial market are ElevateDB, Interbase, Oracle Berkley DB, SolidDB etc.

4. Conclusion

Embedded databases differ from typical databases such as DB2, Oracle, and SQL Server in that it is entirely embedded into the application or hardware device in such a way that the user has very little knowledge, if any, of its existence. Users and administrators are now free from the huge tension of installing and maintaining the database because the database is closely bundled with the application and should be self maintaining. Embedded databases are potable in nature and meant to run on many different platforms with various programming interfaces. It also help in reducing engineering and quality assurance cost, eliminate some support cost in setup and implementation. The nature of embedding databases' instruction sets being linked specifically within and for a specific application gives them a small footprint. Because embedded database reduces the instruction set hence it allows them to achieve performance that is hard to beat. The above discussion end with that future is of embedded database. Besides, embedded industry will flourish in the area of automotive, industrial, consumer electronics in coming year.

REFERENCES

[1] Microsoft Encarta Library 2005.
[2] Graves, Steve. "COTS Databases For Embedded Systems", *Embedded Computing Design* magazine, January, 2007. Retrieved on August 13, 2008.
[4] TinyDB: http://telegraph.cs.berkeley.edu/tinydb/.
[5] http://msdn.microsoft.com/en-us/library/ff647179.aspx (online)
[6] Ramachandra Budihal, Emerging trends in embedded systems and applications (online) *http://www.eetimes.com/discussion/other/4204667/Emerging-trends-in-embedded-systems-and-applications*