SQL Server Mobile – A Database for Pocket PCs and Smartphones

Sitaram Raju

Microsoft India (R&D) Pvt. Ltd.
Gachibowli
Hyderabad
India
sitaramr@microsoft.com

Abstract

In this paper we describe Microsoft SQL Server 2005 Mobile Edition (SQL Server Mobile or SQL Mobile for short), which is a small footprint database for Windows Mobile devices such as Pocket PCs and Smartphones. SQL Server Mobile delivers necessary relational database using a subset of SQL. It has a robust data store with atomicity, concurrency, isolation and durability (ACID) support, an optimizing query processor that uses statistics to create cost-based query plans, and reliable, scalable connectivity capabilities with a backend SQL Server. We present the architecture of SQL Server Mobile and describe some of the tradeoffs that were made to fit the constrained requirements of mobile devices. Finally, we describe some potential future directions for SQL Server Mobile.

1. Introduction

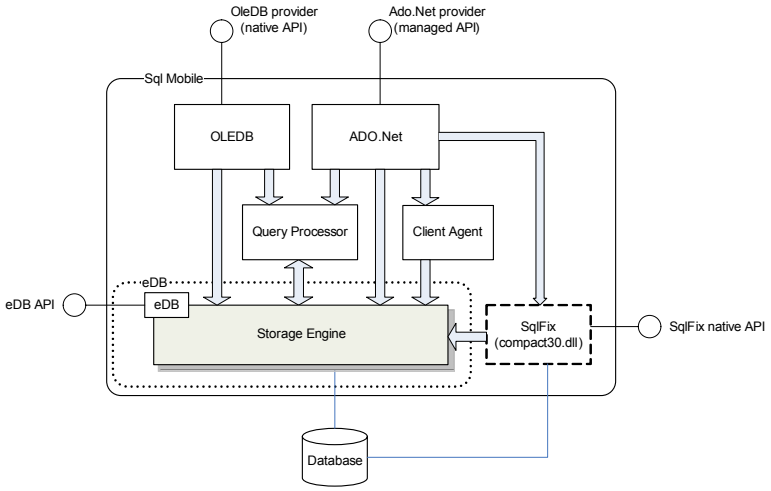
Microsoft SQL Server 2005 Mobile Edition (SQL Server Mobile) is a small footprint database for developing applications that extend enterprise data management capabilities to mobile devices. SQL Server Mobile supports Structured Query Language (SQL) syntax and provides a development model and API consistent with Microsoft SQL Server.

The SQL Server Mobile engine exposes relational database features, such as an optimizing query processor, support for transactions and assorted data types. Remote data access and merge replication ensure that data from a backend SQL Server databases is delivered reliably, can be manipulated offline, and can be synchronized later to the backend server.

SQL Mobile integrates with the Microsoft .NET Compact Framework and Microsoft Visual Studio® 2005, allowing developers to build applications in mangaed code with offline data management capability for disconnected scenarios development for smart devices.

2. Architecture

The diagram below shows the architecture of SQL Server Mobile. There are two APIs to access the database functionality, one is OLEDB which used by native applications like C/C++ and the second is ADO.Net which is used by managed applications like C# and VB.NET.



International Conference on Management of Data
COMAD 2005b, Hyderabad, India, December 20-22, 2005

2.1 Storage Engine

The storage engine is the lowest layer in SQL Mobile stack, it's the component that directly interacts with the database file and provides a logical view of the persisted schema and data. The storage engine provides the following services:

- Exposes operations to create, update and delete records.
- Exposes cursors based on a base table with an optional order i.e. index.
- Provides full ACID transactions (Atomic, Consistent, Isolated, Durable). It uses shadow paging and support for pessimistic concurrency thru locking.
- Support for creating, maintaining and traversing indexes thru B-Trees.
- Change tracking mechanism used by replication.
- Maintains and enforces database constraints i.e. primarily keys and foreign keys.

Coordination between multiple connections to the same database from different processes is done through a main shared memory section and a number of shared memory sections for the locks pool.

2.2 Query Processor

The query processor transforms a declarative SQL statement into a sequence of operations that produces the result for the SQL statement. An optimizer is the heart of the query processor. It employs a combination of heuristic and cost-based transformations to optimize queries. Heuristic transformations rewrite the query into a semantically equivalent form that is expected to lead to a better execution plan. Heuristic transformations are based on syntax and they are always applied when applicable.

One rewrite rule merges multiple adjacent inner join operators, so that the cost-based optimizer can consider other join orders as alternatives to that declared. For example:

```
... FROM (Table_1 INNER JOIN Table_2 ON
Table_1.Col = Table_2.Col)
INNER JOIN Table_3 ON Table_1.Col = Table_3.Col ...
```

is rewritten into:

```
... FROM Table_1, Table_2, Table_3 WHERE
Table_1.Col = Table_2.Col AND
Table_1.Col = Table_3.Col ...
```

This transformation allows the optimizer to consider join orders other than Table_1 joining Table_2, and then joining Table_3.

The application of heuristic transformations is un-conditional. On the other hand, cost-based transformations work differently because these transformations might or might not lead to better execution plans; they are evaluated based on their costs. Cost is the estimated amount of time to run an execution plan and it consists of two parts: Input/Output (I/O) cost and CPU cost.

The following are some examples of operations determined by the cost-based optimizer:

1. Base table scan type (file scan versus index scan)
2. What indexes, if any, are used
3. Join order and join algorithm

SQL Mobile provides a graphical view of query plans, which enable developers to easily see the plan and then modify query hints to quickly achieve performance tuning results.

2.3 Replication

SQL Server Mobile performs synchronization by establishing an HTTP/HTTPS connection to a backend Microsoft SQL Server through Microsoft Internet Information Server (IIS). It uses the authentication, authorization, and encryption services of IIS and can communicate with a SQL Server system located behind a firewall or proxy server. Replication can be performed over wired and wireless local area networks (LANs) such as WiFi, and wide area networks (WANs) such as GPRS. Compression reduces the amount of transmitted data and encryption safeguards sensitive user data. Following a communication failure, transmissions resume from the last successful transmission.

Devices without network interface cards can also synchronize data with a backend SQL Server through Microsoft ActiveSync connection and the use of IIS on the desktop PC which serves as a pass-through conduit to the back-end server.

SQL Mobile supports two options for data replication: Merge Replication and Remote Data Access (RDA). Merge replication enables autonomous data updates on the mobile device and on the server. SQL Mobile provides support for conflict resolution. Usage scenarios for merge replication include read-only replication; data capture and upload; and replicate, update, and synchronize. The database administrator can use both horizontal and vertical filters to define, maintain, and secure unique subsets of data for different clients or groups or clients. Horizontal filters can be used to replicate a subset of rows in a published table. Vertical filters can be used to replicate a subset of the columns in a published table.

In RDA application can submit a SQL query that returns a row set. The resulting row set is transmitted to the mobile device where it is stored in a table. All changes made by the application will be tracked, and at the request of the application, the updated rows will be sent back to the server where they are applied to the SQL Server database i.e. there is no conflict resolution.

3. Database Footprint

The following table shows the sizes of the most important DLLs of SQL Mobile for the ARM v4i processor. As the table shows there is a lot of rich functionality provided in a small footprint.

DLL	Description	Size in KBs
sqlcese30.dll	Storage engine	410
sqlceqp30.dll	Query Processor	874
sqlceca30.dll	Client side connectivity agent	447
sqlceoledb30.dll	OLEDB provider	206
system.data.sqlserverce.dll	Managed provider	223

4. Future Work

SQL Mobile is currently shipping as part of Microsoft SQL Server 2005®. We are currently planning the feature set for the next version of SQL Mobile.

One category of extensions is to add features to SQL Mobile that are found in larger and more mature databases like SQL Server. These features include stored procedures and triggers. There are many columns types in SQL Server that are not available in SQL Mobile such as XML datatype and Common Runtime Language (CLR) integration. Adding these columns and types will reduce the impedance mismatch between SQL Mobile and SQL Server thereby improving developer productivity.

Another interesting avenue for growth for SQL Mobile is the embedded database market. Many next generation applications depend on database mechanisms (like query, indexing, and persistence) and these must be embedded into the application itself. Such embedded database applications are often downloadable and therefore it is important to minimize their footprint. We are investigation added a few features to SQL Mobile, such as in-memory database support, and then positioning it for the embedded database space.