SequoiaDB: The Rise of Modern Data Management in China FinTech

Distributed Architecture Accelerate Fintech Transformation
Executive Summary

Expanding financial services directly to private business and individuals becomes the top priority in China financial industry. The mobile-based technology is growing rapidly to supplement the traditional financial system.

The emerging “FinTech” such as Alipay and TenPay leads the customer engagement into a new level. Each of them probably operates the third-party payment services for over 475 million users. Under these pressure, the traditional banks embrace the modern technology profoundly and gradually to be competitive. Smart banking becomes the phenomenon that banks are in a transition from systematic to intelligent. The expectations of the banking industry are:

- Data security. None of the business and customer data shall be breached;
- Deliver all the financial service to major devices, particularly mobile phones;
- Reduce the IT and maintenance cost;
- Business continuity.

About the measure of size, the challenges are significant. Instead of hiring thousands of developers and engineers as those new “FinTech” companies did, banks are buying the products and solutions. From a vendor’s perspective, the products and solutions shall be differentiated as “enterprise class”, that it is a full package including security, support, productivity, ability to deploy in multiple use-cases, IT complexity reduction, integration capability, and policy management. To achieve the enterprise class, the vendor must own the code and have full control the products.

SequoiaDB, the enterprise distributed database was developed under this circumstance. The majority of the team are from IBM DB2, SAP, and Huawei. With the heritage of the RDBMS development and long time of data management experience, they have built the product from ground zero along with the banking industry evolution.
Data Management Technology Evolution

The banking industry has led the enterprise-class technology for over 30 years in China. The banking systems have evolved along with the regulations on every aspect created by government in decades. Even though banking is a technology-driven business, the data management evolution is a long journey to manage the complexity of the business requirements and regulations and be compatible with the legacy systems at the same time. It shows the critical capabilities for the modern data management in banking industry are mainly in categories as below:

- Scalability and performance
- Distributed Object Storage
- High availability and disaster recovery
- Hybrid Transactional/analytical processing (HTAP)
- In-DBMS analytical capabilities
- Multi-model data management
- Distributed OLTP capabilities

SequoiaDB is customer oriented and rises gradually along with the customer needs. The roadmap of SequoiaDB is the same as the China banking industry evolution journey.

Scalability and Performance

RDBMS is the key to core banking systems. Oracle and IBM DB2 have been extremely reliable for this purpose. They have been dominant of the online transaction processing since the first day. However, RDBMS is designed to run on a single server to maintain the data integrity with ACID support. They are not designed for scale.

For example, banks were used to provide historical transaction search in only recent three months, maximum one year, for customers. Customers had to download the older transactions in the pre-processed document format such as pdf file so that the transaction data was kept in a controllable size by moving the history data in the offline backup disks. The search operation was compromised to yield for the critical core transaction processing since they were running on the same RDBMS.

The similar scenario is in e-banking systems. Customers run many operations such as browsing and searching products, comparing the price, reading the reviews, before they decide to buy. It means data queries are at least ten times more than data inserts or updates. A large number of queries slow down the transaction processing since they are all in OLTP and ACID. However, expanding the RDBMS for these queries are very complicated and costly.

NoSQL was designed in the distributed architecture to scale and achieve performance by compromising ACID support. It separates most of the queries out of the OLTP and keeps the core system healthy. SequoiaDB introduced its first version of distributed NoSQL database in 2012. The performance was ten times faster than the existing RDBMS in a large scale. In 10 billion rows of data volume, response time was within 100ms under thousands of concurrent requests. It also proves its robustness and automated scalability by testing over 1000 nodes. Therefore, SequoiaDB had been adopted in China banking industry quickly.

Soon after, the SequoiaDB team found the side-effects from NoSQL that developers were not very satisfied without transaction and SQL support. The SQL and transaction support was added in SequoiaDB v2 responsively. Tunable consistency and multi-model support became the standard feature.
Object Storage and Content Management

Financial Services find themselves in challenges to handle a large amount of unstructured data in different types such as pictures, videos, and files. Enterprise Content Management (ECM) solutions such as IBM CM8, FileNet, Documentum are used for a long time. Same as RDBMS, they are not designed to scale. Furthermore, they can be replaced easily comparing to RDBMS.

People started to build their in-house platforms with database integrated with file-level storage, i.e., Network-attached storage (NAS). They use the database for meta data point to the file location on the NAS. This solution is deprecating later on due to its data inconsistency, scalability, and expensive cost.

HDFS is brought up since Hadoop became a standard setup for Big Data. People started using HDFS as storage for ECM. This solution is fine for archiving purpose, but it is not designed for high concurrency requests.

Real-time object data access is a new trend. For example, Banking and insurance industries in China are first to adopt the facial recognition for supporting account open and security verification scenarios. Based on the single Massively Parallel Processing (MPP) architecture, SequoiaDB built a service layer on top the SequoiaDB object storage engine as a new distributed ECM competitor on the market, which is called SequoiaCM.

It has been proved very successful to penetrate the banking industry including tens of top 50 banks. Many traditional ECM vendors in China integrated with SequoiaCM to enhance the existing solutions.

GDPS and Active-Active Disaster Recovery

High availability is the mantra of the day. China Banking Regulation Commission has required all banks data centers to be Geographically Dispersed Parallel Sysplex (GDPS) capable, which means distributed data centers are required as standard deployment. Distributed DBMS will be the top priority for the data management modernization.

Most of the banking data centers also require active-active disaster recovery capability, also known as dual active data centers which combines both High Availability and Disaster Recovery in the approach to data storage, data processing, and data recovery. Data management can be expected to achieve continuous availability at a lower cost and to maximize the use of data centers with the least amount of effort.

SequoiaDB uses MPP architecture to serve GDPS natively. It has the flexibility to be deployed based on the customer High Availability requirements. SequoiaDB supports the dual active data centers depending on the network capacity. SequoiaDB deployment for dual data centers in the same city works perfectly. But it’s hard if the data centers are far away, for instance, data centers in different cities.

<table>
<thead>
<tr>
<th>FIGURE 1</th>
<th>SequoiaCM Specification Comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Solutions</strong></td>
<td><strong>Traditional CM (EMC / IBM)</strong></td>
</tr>
<tr>
<td>1. Type</td>
<td>Archiving</td>
</tr>
<tr>
<td>2. Market Size</td>
<td>Large (Shrinking dramatically)</td>
</tr>
<tr>
<td>Version Control</td>
<td>Yes</td>
</tr>
<tr>
<td>Batch Control</td>
<td>Yes</td>
</tr>
<tr>
<td>Meta Model</td>
<td>Yes</td>
</tr>
<tr>
<td>Check In/Out</td>
<td>Yes</td>
</tr>
<tr>
<td>Life Cycle Management</td>
<td>Yes</td>
</tr>
<tr>
<td>Storage and File System</td>
<td>Multiple Storage Types without FS</td>
</tr>
<tr>
<td>APIs</td>
<td>Many</td>
</tr>
<tr>
<td>Content Processing</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td><strong>Distributed File System (Risks on Data Replication and Recovery)</strong></td>
</tr>
<tr>
<td>Meta Data on Content Management</td>
<td>Meta Data in RDBMS, data in Object Storage</td>
</tr>
<tr>
<td>Distributed Architecture</td>
<td>Difficult</td>
</tr>
<tr>
<td>High Availability</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td><strong>High Concurrency</strong></td>
</tr>
<tr>
<td>API Protocol</td>
<td>Many</td>
</tr>
<tr>
<td>File Access API</td>
<td>Built-in APIs</td>
</tr>
<tr>
<td>Function API</td>
<td>Many</td>
</tr>
<tr>
<td><strong>ROI</strong></td>
<td><strong>Expensive Hardware &amp; Maintenance</strong></td>
</tr>
</tbody>
</table>

Source: SequoiaDB
Data Centralization (Operational Data Lake) and HTAP

Banking systems were built independently. Each of them maintains their own data in RDBMS. Business requirements sometimes need to have multiple service calls to these systems to retrieve the data in pieces and then merge them into a new data result. For example, personal statement in private banking is data summary that comes from over 40 core systems such as debit, credit, and investment. This kind of low-efficient data processing would not make it possible to be a general function for customers - the end users. It would be a disaster to handle such concurrent requests across over such amount of core systems.

Data centralization, or data lake, is the answer for it. A data lake is the single data central repository to handle the high concurrency data processing. The data in the data lake is from different sources such transaction data, behavioral data, logging data, and so on. Transactional data is synchronized by database bin logs. Behavioral data and logging data can be either send by restful API or log extract tools. The operational DBMS for data lake must be ETL (Extract, Transform, Load) friendly.

There are many scenarios people want to have in business real-time data monitoring, data report and decision making. The other purpose for a data lake is hybrid transactional/analytical processing (HTAP).

Traditional architect separates the OLTP and OLAP. The gap causes difficulties over data consistency, data platforms, and skills to become large hurdles for an enterprise to adopt a new data management solution. RDBMS and Big Data (Hadoop) became the two sides of the data processing. People chose the operational then have to give up the analytical, vice versa.

The operational DBMS is then evolving, with new, innovative entrants and incumbents supporting the in-DBMS analytical capability. SequoiaDB has its Spark connector to integrate with Spark. It can be used as a data source of Spark and support Spark SQL.
SequoiaDB data analytics solution supports two scenarios - pre-defined requests and ad-hoc requests. Pre-defined requests are easy to handle, either from offline batch data processing or real-time data streaming processing.

Ad-hoc Query is very different since there is no way to know what the ad-hoc SQL is and what data related, so SequoiaDB has to use a scheduling tool to find and retrieve the data on-the-fly. A sandbox has to be introduced for the ad-hoc test run to avoid the destructive requests.

**Distributed OLTP**

Data scale is increasing dramatically nowadays. The banking core systems are required to keep up with the pressure, which means RDBMS needs to be scalable and distributed. For example, people started working on a kind of middleware on top of RDBMS like MySQL to improve the parallel processing. Therefore, distributed online transaction processing is required when transactions need to cross multiple data nodes or instances.

A distributed transaction must be synchronized and provides full ACID support. In the application level, the distributed OLTP is transparent to the developers. Developers don’t need to change any code to be distributed, and the transactions are managed the same as the local transactions.

SequoiaDB is in progress in the other direction. It is improving its OLTP support in native MPP architecture. SequoiaDB 3.0 will fully support the distributed OLTP.

**Conclusion**

While traditional RDBMS is still a common choice in enterprise-class data management, demands of high performance and reliability at competitive cost grows increasingly for modernizing legacy systems. The next generation DBMS is built for distributed OLTP, hybrid transactional/analytical processing (HTAP), and distributed object storage in a large scale to help modernize the data management in China.

IT spending in the banking ranks to the top in all industries of China. Hence, SequoiaDB, an enterprise-class distributed database management platform proves its success by penetrating the vertical market sector, Financial Services, especially banking industry dramatically.
Author Introduction

Tao Wang, Co-founder and CTO of SequoiaDB

Tao is in charge of the company’s strategic planning, and is responsible for the management of the entire database R&D. Tao has worked at the IBM Toronto Labs (DB2 UDB Development Lab) as a member of the IBM DB2 Global Top Technical Panel.

In 2012 Tao has finished the first version of the database, and founded the company as co-founder and CTO. During the past 6 years, with Tao as the technical leader, SequoiaDB has become the leading database vendor in China and SequoiaDB is now penetrating the vertical sector financial industry quickly including tens of top 50 banking clients.

Company Introduction

SequoiaDB, a China company, is an enterprise-class distributed database management platform built for traditional OLTP, hybrid transactional/analytical processing (HTAP), and object storage in a large scale. Both their co-founders are originally from IBM DB2 Toronto Lab. The majority of the team are from IBM DB2, SAP, and Huawei. With the heritage of the RDBMS development and long time of data management experience, they have built the product from ground zero along with the banking industry evolution. SequoiaDB has become the top distributed database vendor in China.

SequoiaDB has two major products:
- Enterprise Distributed Database -- SequoiaDB
- Distributed Content Management Software -- SequoiaCM

SequoiaDB is now penetrating the vertical sector financial industry quickly and had tens of Top 50 banking clients in China. The company has received Red Herring Global 100 reward in two consecutive years. SequoiaDB was also recognized as one of the 50 most innovative companies in China by Fast Company.
While relational DBMS choices are common in IT modernization decisions, new data management options offer high performance and reliability at increasingly competitive price points. This note introduces senior IT leaders to modern data solutions for modernizing legacy systems.

Key Findings

- Emerging data management options such as NoSQL, Apache Hadoop and in-memory database management systems (DBMSs) offer new modernization choices for a variety of data challenges and deployment scenarios.
- The falling cost of memory, coupled with the need for faster business insights, is driving increased use of in-memory computing solutions for high-performance workloads.
- Confusion over data consistency and skills are the largest hurdles when adopting new data management solutions for IT modernization efforts.

Recommendations

For senior IT leaders embarking on IT modernization:

- Start your modernization efforts by renovating first, using new data management products to add features and capabilities to existing environments. Target features that may be cost-prohibitive or operationally difficult in your legacy environment.
- Use a NoSQL DBMS when the requirements of the application allow for relaxed consistency models and/or when geodistribution of data is required.
- Exploit in-memory options — such as DBMSs, data grids and event-processing platforms — to meet scalability and low latency transaction processing and analytics requirements.
- Align modern data management solutions with your service-level agreements (SLAs) and expected data interaction patterns.

- Target small-scope proofs of concept to understand the operational and data consistency characteristics of new options.

Analysis

Start Your Renovation With New Opportunities

The decision to modernize application systems is often driven by such factors as rising costs, lack of skills and an increasing need for business agility. Regardless of the reason, modernizing customized legacy systems requires a change to one or all of the technologies used to implement the application — runtime platform, programming language, and DBMS or data storage approach. These three technologies can be modernized together or, in some cases, independently. When migrating a complete, customized legacy application from one platform to another — whether via a complete rewrite, code transformation or package migration — all aspects of the application will change at once.

Several common modernization decisions affect the underlying data structure. Most frequently, the data migration issues associated with custom-made legacy applications are related to moving from prerelational data types and structures to relational database management systems (RDBMSs).

Historically, the three most common destinations we hear about from our client base when migrating prerelational data stores associated with custom-developed legacy systems are IBM DB2, Microsoft SQL Server and Oracle Database. And for good reason — the practices used to migrate from prerelational to relational are well defined. The most common data stores associated with these legacy applications are sequential or indexed file structures, as well as prerelational DBMS (for example, networked, hierarchical or inverted list) database structures.

However, modernization efforts are no longer limited to traditional, even disk-based, RDBMSs. Several new options are available, including:

- NoSQL DBMSs characterized by eventual consistency models, geodistribution of data and flexible data schemas.
- NewSQL DBMSs offering familiar atomicity, consistency, isolation and durability (ACID) guarantees with horizontally scalable deployment options, typically characterized by NoSQL products.
- In-memory DBMSs for high-performance applications and analytics.

Today’s modernizations efforts face more challenges than just a change in data management platform. Mobile applications allow data to be managed anywhere, not just from a centralized location. Data may need to be geographically distributed to comply with data sovereignty laws, or to insure against a single point of failure in the face of natural disasters. SLAs may change, requiring a faster time to value for new data. Finally, transactional data or analytical results may be spread across cloud and on-premises environments. Modern data solutions address each of these challenges.

Recommendation:

Start your modernization efforts by renovating first, using new data management products to add features and capabilities to existing environments. Target features that may be cost-prohibitive or operationally difficult in your legacy environment.

NoSQL and NewSQL as Modernization Alternatives

NoSQL DBMSs are relatively new and there are several varieties, making it difficult to determine the best one for your particular requirements. NoSQL DBMSs can be arranged by type, which will make them easier to evaluate (see Note 1).

Gartner has published two documents that provide guidance for selecting the type of NoSQL DBMS that is best for your use case: “Decision Point for Selecting the Right NoSQL Database” and “Decision Point for Selecting the Right Database Format: Relational, XML or NoSQL.” Example use cases of NoSQL.
DBMSs can be found in “A Tour of NoSQL in Eight Use Cases.”

When it comes to modernizing legacy systems, NoSQL can be used to replicate traditional transactional data in such a way that it can be quickly and easily accessed for higher volume, read-only requirements, perhaps driven by increased mobile access and geographic distribution of workers. For example, several consumer banking companies use NoSQL DBMSs to present transaction information to online banking users. This removes some load from the traditional DBMS environment, reducing processing demands and, therefore, cost.

Another approach that has recently surfaced concerns a group of DBMS products called NewSQL. Typically, these offer the advantages of NoSQL such as new, more flexible data models and geodistribution of data while enforcing ACID consistency. Although these models have greater latency for transactions than NoSQL DBMSs, generally they do offer superior performance for geodistribution of data over the distributed two-phase commit used by most RDBMS products.

**Recommendation:**

Use NoSQL DBMS when the requirements of the application allow for relaxed consistency models or when geodistribution of data is required.

**In-Memory Capabilities Proliferate**

The growing sizes and falling price of DRAM has driven the proliferation of in-memory technologies. These range from new in-memory DBMSs (IMDBMS) to the addition of in-memory capabilities on formerly disk-only products. An IMDBMS stores the entire database content in-memory and accesses it without having to perform disk input/output (I/O) operations. Flash or disk is used to ensure no data is lost, or to provide additional capacity if the data exceeds the available memory. IMDBMS is not limited to RDBMS only, as many NoSQL DBMS products also make use of in-memory computing, from partial IMDBMS and integrated in-memory data grids to full IMDBMS. The major advantages of IMDBMS implementation include high-speed performance, scalability and potential for consolidation.

Today, IMDBMS features are typically optimized for analytical or operational use cases:

- **Operational features are used for transactions where all necessary data is stored in-memory (DRAM) on a server. All DBMS operations, such as select, update and delete, are performed in-memory. An operational IMDBMS can scale vertically (in a single server) or horizontally (in a cluster of servers). It has all the necessary structures in-memory and is not simply an in-memory disk-block cache.**

  Similar to the scenario using NoSQL for modernization, IMDBMSs may provide another alternative for high-volume access to read-only data. They can also be used to improve performance on large-volume data environments in a transactional environment, since data loss can be eliminated and transactional integrity can be maintained.

**Recommendation:**

Exploit in-memory options — such as DBMSs, data grids and event-processing platforms — to meet scalability and low-latency transaction processing and analytics requirements.

**Data Consistency — Changing the Logical Unit of Work**

Ensuring the integrity of the logical unit of work has, historically, been managed by the transaction monitor and/or the DBMS. All changes to data are strictly consistent within the context of a transaction, and all distributed nodes receive the updates. This guarantee of consistent data comes at the cost of throughput, due in part to the ACID consistency model of most RDBMSs. NoSQL DBMS applications take a different approach to this problem by ignoring it. NoSQL DBMSs fargo the logical unit of work concept, they don’t have transaction boundaries. Instead, NoSQL DBMSs rely on application developers to define and maintain those boundaries. This is important because most NoSQL DBMSs use eventual consistency to achieve higher levels of throughput. Eventual consistency propagates data changes opportunistically. Data becomes consistent throughout the cluster over time. That time might be milliseconds, minutes or longer depending on factors such as cluster utilization, network characteristics and potential network downtime.

If eventual consistency is acceptable for the particular application, then this approach is fine. However, many legacy applications and many legacy programmers are accustomed to the notion that data integrity is maintained at all times — either the entire update works, or none of the update works. Consistency is critical for many legacy transactional systems and is expected to be managed by the application and/or application runtime environment. New applications may use NoSQL, accepting eventual consistency as satisfactory, whereas other, more time-sensitive applications may require immediate consistency within the DBMS.

**Recommendation:**

Understand the trade-offs between performance and eventual consistency in newer DBMS platforms before committing to their use.

**Polyglot and Multimodel DBMS — When One DBMS Just Won’t Do**

Given the heterogeneous nature of today’s technology implementations and the diverse nature of current application demands, a single data type may not meet all of your requirements. Multiple data types may be required. For example, modernization use cases may require representing data as a graph as well as relationally. This mixing of data types using dedicated DBMSs is called polyglot persistence. It is increasingly popular, but not without its challenges.

Creating multiple versions of a data store can be seen as problematic, since multiple versions of “the truth” are inconsistent with...
many traditional legacy application and DBMS styles. Creating multiple versions does introduce the problem of synchronizing the data among the different data stores. This synchronization could be done in real time, or at some regular schedule — hourly, daily or weekly. The schedule depends on the latency demands of the data. If the data does not have to be continuously synchronized, longer synchronization cycles may be acceptable.

Another challenge with a polyglot approach is the complexity of the operating environment. Multiple DBMSs must be deployed, configured and optimized, with little similarity between products. The availability of skills in a diverse environment remains the largest hurdle for polyglot deployments.

The next iteration of polyglot persistence is multimodel DBMS (see Figure 1). Multimodel, as the name implies, is the support of several data types in a single DBMS platform. Just as traditional DBMS vendors included features and data types from XML and object-oriented DBMSs, vendors now also include JSON, graph and key-value capabilities. A multimodel approach allows organizations to reuse skills and existing vendor relationships for modernization efforts. Multimodel isn't limited to traditional vendors. Emerging DBMS vendors are extending beyond their original products and including new data types. Graph is the most commonly added, alongside table-style and document stores.

**Recommendation:**
Align modern data management solutions with your SLAs and expected data interaction patterns. Target small-scope proofs of concept to understand the operational and data consistency characteristics of new options.

**Evidence**
The fact base for this research comes from analysis of Gartner’s inquiry service from January 2015 through March 2016, and primary research conducted by the analysts.

---

**FIGURE 1**

Polyglot Versus Multimodel

![Diagram of Polyglot Versus Multimodel](image)

**Note 1. NoSQL DBMS Types**
The NoSQL market consists of four types of DBMSs, grouped by the type of data model supported:

1. **Document-store DBMS** — Document-store DBMSs hold data in a hierarchical, tree-like format. Document stores do not enforce an externally defined schema, making them an ideal choice when combining data from multiple sources. Document stores commonly describe data using web-centric interchange formats like JSON or XML. These formats allow for easy mapping to web applications, making document-store DBMSs popular for rapid application development.

2. **Graph DBMS** — Graph DBMSs store information in a structure that records the direct relationship between any two adjacent elements. Nodes have properties and connect to other nodes at “edges.” This inherent structure makes graph DBMSs ideal for storing and analyzing connected data for use in relationship analysis, route planning and optimization, as well as identity and access control (among others).

3. **Key-value DBMS** — Key-value DBMSs store both the key and value as binary objects. Dating back to IBM’s Indexed Sequential Access Method (ISAM) in the 1970s, key-value is actually the oldest NoSQL model. Key-value DBMSs evolved to support rapid scaling for simple data collections by automating the process of distributing data across several nodes. Their attributes allow key-value DBMSs to support data access patterns, largely driven by key lookups and requiring consistent access times. They are commonly used in scenarios requiring a constant stream of small reads and writes, such as web cookies or session tokens.

4. **Table-style DBMS** — Table-style DBMSs store rows of data in tables, making them the most similar in concept to the relational model. However, table-style DBMSs do not have relationships between rows and they support flexible schema definitions. These traits make table-style DBMSs popular for storing semistructured data, such as log or clickstream data.

Gartner Research Note G00275422, Dale Vecchio, Nick Heudecker, 07 June 2016
SequoiaDB, a China company, is an enterprise-class distributed database management platform built for traditional OLTP, hybrid transactional/analytical processing (HTAP), and object storage in a large scale. Both their co-founders are originally from IBM DB2 Toronto Lab. SequoiaDB has become the top distributed database vendor in China. They are penetrating the vertical sector Financial Industry quickly and had over 30 banking clients. The company has received Red Herring Global 100 reward in two consecutive years. SequoiaDB was also recognized as one of the 50 most innovative companies in China by Fast Company.