MCloudDB: A Mobile Cloud Database Service Framework

Lihui Lei
School of Computer Science
Shaanxi Normal University
Xi'an, China
leilihui@snnu.edu.cn

Sabyasachi Sengupta, Tarini Pattanaik, Jerry Gao
Computer Engineering Department
San Jose State University
CA, USA
jerry.gao@sjsu.edu

Abstract — The recent advance in mobile network technologies (3G/4G) provides an efficient mobile network infrastructure supporting mobile users to access large volumes of mobile data. Now, many mobile applications are developed based on mobile databases on devices and conventional databases. Due to the limitations of mobile devices, using mobile databases on mobile devices encounters certain scalability issues in mobile data accesses and storage. One alternative approach is to use cloud-based databases to support mobile users and applications on mobile devices. This also runs into the several challenges, such as mobility support, localization, energy consumption, and performance concerns. This paper presents a mobile enabled cloud database solution, called MCloudDB, to address the related issues and needs. MCloudDB provides a mobile enabled cloud database infrastructure with a framework, which can be used as a bridge to connect and integrate mobile applications with back-end cloud databases to support mobile cloud data management and access services. The paper presents the design of this framework and the related key solutions for essential features in mobile cloud databases, such as multi-tenancy, elasticity, seamless connectivity and disaster recovery. A simple application example and related experiment results are reported.

Keywords—Mobile Computing; Mobile Cloud; Mobile Cloud Databases; Mobile Databases; Cloud Computing

I. INTRODUCTION [1]

As mobile devices continue to proliferate, Gartner predicts an increased emphasis on serving the needs of the mobile user in diverse contexts and environments, as opposed to focusing on devices alone. As listed by Gartner[2], ***computing everywhere*** has been identified as one of top 10 technology trends for 2015.

Supporting the fast increasing needs of mobile data accesses and mobile computing services, new mobile enabled database solutions and systems are needed due to the following reasons. Firstly, existing mobile DB technologies on mobile devices have limited data storage size and computing power. Hence, today software engineers are tempted to build mobile applications using a mobile database solution in a hierarchical database model by leveraging a centralize database system. In a hierarchical database model, a mobile database system requires a centralized storage server to host the master copy of data, and mobile clients to contain their own copies of mobile data. However, different mobile client groups may have different requirements on mobile data schema, which requires multi-tenancy in mobile database access services. In addition, current mobile database solutions also encounter other issues and challenges in supporting mobile user accesses. Typical ones include unreliable mobile connectivity, mobile data access performance, and data liveliness of mobile applications. Moreover, frequent connection and disconnection of mobile devices also brings the mobile DB transaction processing challenge and mobile data synchronization needs. The big challenge is the mobile scalability for mobile data access services.

Fortunately, cloud databases can be utilized to address the first issue. With the fast adoption of cloud computing in many enterprises, software vendors started to deploy mobile applications in the third-party cloud infrastructures to support mobile data storage, management, and accesses. This cloud-based approach not only provides elastic scalability for mobile applications at the infrastructural level, but also provides an unlimited elastic mobile data storage.

Secondly, using this approach based on current cloud databases have some limitations in supporting certain ideal mobile database features, such as mobility-based data transactions, easy and seamless integration and connectivity with mobile application servers, and reliable mobile data access transactions with high tolerance for connection loss, and recoverable service sessions coping with frequent disconnections from mobile devices and users.

The research work reported in this paper aims to target to these issues and needs by introducing a new mobile enabled cloud database solution, known as MCloudDB. It provides a mobile enabled cloud database service framework which can be used as bridge to connect and integrate mobile applications with a back-end cloud database to support unlimited mobile data access services. Software engineers are able to utilize this framework to build mobile applications supporting scalable cloud-based mobile data storage and management, multi-tenant-based mobile data access sessions, and reliable mobile connectivity. In addition, MCloudDB provides the common implementation framework to support highly granular multi-tenant database management transactions and operations.

This paper is organized as follows. Section 2 reviews the related work, and compares the existing work and our proposed MCloudDB - a framework for a mobile cloud data solution. Section 3 presents the infrastructure and design of MCloudDB. It can be used by engineers as a framework to build mobile enabled cloud database services for mobile applications.

Supported by Funding of China Scholarship Council and National Natural Science Foundation of China (No.61003061, No.11271237)

[2] http://www.thecloudnetwork.net/content/gartner-identifies-top-10-strategic-technology-trends-2015

Section 4 presents a case study for its application usage. The final conclusion and future work are discussed in Section 5.

II. RELATED WORK

This section reviews the related work from two different aspects: (a) mobile databases, and (b) mobile cloud databases.

A. *Mobile Databases*

A mobile database refers to a database resides on a mobile device. Supporting mobile data services of mobile applications require a mobile accessible database solution [1, 2], which is the union and integration of a central controlled distributed database and a disconnected database on mobile devices. Unlike conventional database systems, there are special features and needs for mobile databases, including mobility, location awareness, mobile transaction management, query process, data replication, data synchronization and caching.

- ***Mobility and Location Awareness***

Mobility refers to the ability to change user locations while retaining network connection, which is one key motivation and service capability in mobile computing. In mobile computing, there are two common mobility features [3]. The first one is the micro mobility, which concerns the movements within a geographically contiguous area where mobile user remains in the same wireless network. The other is the macro mobility, which refers to the wide area mobility that could involves heterogeneous networks. At any time, mobile users can connect or disconnect its mobile devices at different places.

Location awareness, the ability to determine geographical position, is an emerging technology with significant benefits and important privacy implications for mobile devices users. The problems on location awareness have motivated an intense research in mobile data services. Among these services, location based services (LBS) and location dependent queries have attracted a lot of attention from both industry and research communitys [4]. Primary location-based services include communication, visualization, navigation, tracking, search and billing services. These services provide the foundation supporting more complex applications and services. Research work on locality based data modeling [5] has provided a basic foundation.

- ***Mobile Transaction Management & Query Processing***

In mobile computing, a mobile transaction refers to a transaction where at least one mobile host is involved. To cope with mobility, mobile transactions tend to be long-lived even though some transaction failures may occur [6].

Processing long-live transactions for mobile data services need certain transaction management services in a mobile environment to cope with the challenges in preserving data consistency and fault tolerance. As we known mobile database accesses via mobile networks or wireless internet are prone to frequent wireless disconnections, and mobile devices are limited to mobile network bandwidths, computing power and storage limitations, it is always a challenge to support mobility and ensure efficient executions for mobile transactions [7].

Query processing is a key issue in the research of mobile database transaction service. Previous works on query processing of mobile transactions can be found in [8]. An important need of query processing is query optimization. Many execution strategies of query processing are query optimization solutions to reduce the time of processing a query. Query optimization can be done statically before executing the query or dynamically while the query is executed. In a mobile environment, static optimization is inadequate due to mobility. Thus, dynamic optimization or hybrid optimization is more suitable for query processing.

- ***Data Replication/Synchronization/Caching***

Data replication is the process that allows building a distributed environment through managing multiple copies of data and caching one copy on each site. Replicating data at several sites is a powerful mechanism which not only increases performance but also provides fault tolerance for demanding database applications [9].

The major concern of data replication is to keep the consistency of different data copies. Whenever a data update is performed on any data story, the update must be propagated to the other data stores to maintain the consistency. The strategy of propagating the update can be synchronous or asynchronous. Synchronous replication ensures highest level of data integrity that requires the availability of participating sites and transfer bandwidth. If one of the participating sites holding a replica is not ready, then the transaction may not be proceed. Asynchronous replication is more flexible than synchronous replication. For asynchronous replication, database synchronization can be done in various time intervals depending on different applications and service providers. Another benefit of asynchronous replication is that a single site could work even if a remote server is unreachable or down [10, 11].

Besides, data caching plays a key role in mobile databases, because it alleviates the performance limitations of weakly-connected or disconnected operations [12].

B. *Mobile Cloud Databases*

With the recent advance of cloud computing, people start to pay more attention to a new computing paradigm, known as Mobile Cloud Computing (MCC). MCC has an advantage to cope with the limitations of mobile devices by leveraging elastic computing resources in clouds to provide mobile data services and applications for mobile users at anywhere anytime.

As defined in [13], MCC is "an emergent mobile cloud paradigm which leverage mobile computing, networking, and cloud computing to study mobile service models, develop mobile cloud infrastructures, platforms, and service applications for mobile clients." Its primary objective is to delivery location-aware mobile services to users based on scalable mobile cloud resources in networks, computers, storages, and mobile devices. Its major goal is to deliver users with secure mobile cloud resources, service applications, and data using energy-efficient mobile cloud resources in a pay-as-you-use model.

As pointed out in [13], new technologies and solutions are required to meet new demands on mobile data services in mobile cloud computing. Some research work has been done to address some expected key features for cloud databases and mobile databases. For example, the authors in [14,15,16] focus on two important features listed below.

- ***Multi-tenancy***

As discussed in [14], multi-tenancy is a distinct feature of mobile SaaS and mobile clouds. With multi-tenancy, mobile users can access its tenant-based data services at the different levels listed below.

a) ***Physical data store level*** ,which refers the data storage infrastructure, where tenant-based mobile users are able to share, store, and access the same physical data store.
b) ***Logic database level***, which refers to a database schema, where tenant-based mobile users can define and share the same database scheme.
c) ***Data table level***, which refers to shared DB tables among tenants, where database tables can be shared by tenanted users.
d) ***Application level***, where different application functions can be provided for tenanted users. Tenant-based mobile users can share and access the same application logics.
e) ***User interface level***, where diverse tenant-based mobile DB data can be accessed, presented, formatted, and visualized.

- ***Scalability***

Scalability is a desirable property of mobile SaaS applications in mobile clouds. Supporting large-scale mobile data services require a mobile enabled cloud database to support system-level scalability for large-scale mobile data accesses, including scalable data sets, data access transactions, and mobile data transferring. Therefore, scalability for mobile data services can be achieved at three levels.

a) ***Scalability at the Infrastructure level***, where elastic data storage servers are allocated and provided to support scale-out for mobile data stores whenever on-demand provision requests from users are processed.
b) ***Scalability at the Platform level***, where a data service platform is provided with various tools and service programs to support the construction, deployment, hosting, and maintenance of mobile data application services upon user requests.
c) ***Scalability at the SaaS level,*** where mobile data-as-a-services is provided to support large-scale mobile user requests with load balance and scale-out capability.

There are a number of key scalability principles: 1) partitioning by function or grouping functions that are closely related to each other; 2) splitting data horizontally so that data processing can be done in parallel; 3) avoiding distributed transactions; 4) decoupled functions interact one and another asynchronously so that they could be scaled independently; 5) moving to asynchronous processing [15]. Hence, emergent cloud databases introduced new service solutions to support data service scalability. One popular service function is known as data sharding that separates very large databases into smaller, faster, more easily managed parts. The *next* service function is data migration, which provides automatic or semi-automatic data migration. The *third* one is data replication [16].

C. *Existing Mobile Cloud Database Products and Solutions*

Currently, only few commercial mobile cloud database products are available. Table 1 compared four of them. More detailed information can be found in the following URLs.

- database.com(http://www.database.com)
- CloudAnt(https://cloudant.com)
- NuoDB (http://www.nuodb.com)
- Couchbase (http://www.couchbase.com)

Table 1: Comparison of Mobile Cloud Database Products

Attributes	NuoDB	Database.com	Cloud Ant	Couchbase
Company	NuoDB Inc	Sale force	Cloud Ant	Couchbase Inc
Data Model	Relational	Relational	Document based	Document based
Data Transaction	ACID	A transaction is only on an API call to the DB	ACID and eventual consistency	ACID compliant on single item
Client Type	Thin/Fat	Thin/Fat	Thin/Fat	Thin/Fat
Data Access	ODBC/ JDBC/.Net	SOAP/REST	SOAP /JavaScript	REST
Data Republication	P2P based, Partial on demand	Auto	Multi-Master	Multi-Master/ Multi-Slave
Sharding	No Supported	Not Supported	Supported	Supported
Indexing	Yes	Yes	Yes	Yes
MVCC	Yes	No	Yes	Yes
Multi-Tenancy	Yes	Yes	Yes	Yes
Security	Secure Communication /Data encryption	Authentication/ Row-lever Security/ Persimmons	Data encryption/ Access Log/ Communication via HTTPS/ /Authorization	SASL authorization/ Data encryption

MVCC: Multi-Version Concurrency Control

Some of major features of these products are highlighted here.

- ***Data model*** – Two popular data models are used. They are: a) relational based data model; and b) document based data model.
- ***Mobile client*** – There are three types of mobile clients for mobile data accesses. They are: a) ***thin*** client, ***smart*** client, and ***fat*** (or known as ***thick***) client. The listed products only use either ***thin*** client or ***fat*** client.
- ***Mobile data access*** – Several popularly used emergent technologies for mobile data accesses include SOAP, RESTFUL, and OCBD/JDBC and .NET.
- ***Data Replication*** – All of the listed mobile cloud DBs provide certain data replication approaches. Typical approaches include P2P based, on-demand based, master-slave, and multi-master data replication methods.
- ***Data Sharding*** – Among of these four solutions, only CoudAnt and Couchbase provide an auto data sharding solution.

- ***Mobile Security*** – Different security solutions are used by these products. Typical security solutions focus on mobile security in mobile communications, mobile user authentications, and mobile data integrity and security.

This paper introduces a new mobile cloud database solution, known as MCloudDB. It provides a cloud-based mobile database framework, which is used to play as a bridge to connect and integrate mobile applications with back-end data clouds for mobile enabled data access services. Software engineers can use the framework to build mobile applications to achieve several ideal mobile cloud computing features, such as multi-tenancy and elasticity scalability. Meanwhile, it also keeps many mobile computing features, such as mobility and automatic transaction recovery. In the next section, the design and the key components of this framework will be described.

III. MCLOUDDB - MOBILE CLOUD DATABASE SOLUTION

This paper reports our MCloudDB project which is develop to meet the needs of mobile enabled database services for engineers to build desirable mobile data services by leveraging existing cloud databases. MCloudDB is a mobile enabled cloud database solution, which can be used by engineers as a common reusable mobile cloud DB service framework to develop mobile application (or mobile SaaS). Table 2 highlights some of its major features.

Table 2: Key features of MCloudDB

Feature	Value Proposition
Data access abstraction	MCloudDB has a data access abstraction layer, which provides common database access APIs for mobile applications to avoid vendor lock-in. Besides, mobile applications can interface with the right back-end databases based on the user specification and content type.
Mobile data transfer latency reduction	Mobile applications are not required to directly interact with master databases. Instead, they interact with the middle tier to enable abstracts locality of mobile data for mobile applications. This reduces the latency of data transfers and improves user experiences.
Mobile data transfer resiliency	Most clouds or mobile databases do not guarantee transfer control and resiliency, while the middle tier provided by MCloudDB can be used to keep tracking of ongoing mobile data transfers. Meanwhile, the fault recovery capability is provided to resume unfinished transfers or disconnected mobile service sessions.
Data filtering and user profile	MCloudDB allows users to tag their data with respect to their locations, so that mobility parameters can be used in load balance and service request processing. Meanwhile, user mobility data can be cached at the right datacenters in the *Mobile Data Infrastructure Cloud*, which could enhance user experiences when they access data.
Multi-tenancy management	MCloudDB enables mobile applications to have multi-tenancy at various levels (schema-based, table-based and column-based). Most Cloud databases provide either one of these schemes but not all. Supporting the column-based multi-tenancy scheme enhances the utilization of cloud storage and improves the management of databases.
Scalability management	MCloudDB provides scalability solutions to enable integrated mobile applications to support large-scale mobile data services for mobile users.
Database lookup & search management	MCloudDB provides the data indexing capability based on user locations. This feature further improves the efficiency of mobile database lookup and search.

A. *MCloudD Infrastructure*

The high-level infrastructure of MCloudDB is illustrated in Figure 1. This infrastructure has a multi-tier distributed structure that comprises of *Database backend infrastructure tier*, *Database service infrastructure tier*, *Internet tier*, and *Database client tier*.

- In the *Database backend infrastructure tier*, a Backend Database Cloud (BDC) is employed to address the issue that how customer data is actually stored.
- In the *Database service infrastructure tier*, there is an intermediate Mobile Data Infrastructure Cloud (MDIC) deployed on the Top-of-Rack mobile edge routers at the cloud datacenters. MDIC are hosted on virtual machines for performing the tasks embodying the key features, such as multi-tenancy, elasticity, data caching, automated pay-per-use billability and so on. Connectivity between the BDC and the MDIC is achieved through tunneling protocols via dedicated high-speed fiber optic links.
- The *Database client tier* is composed of the MCloudDB based client applications which interact with the MDIC and are installed in clients' mobile devices. Initially, all connections are made between the client applications and the Central MDIC; then the Central MDIC determines which is the best equipped Regional MDIC for serving a client; and finally each client application and the selected Regional MDIC are connected.
- *Internet Tier* between the Database client tier and the Database service infrastructure tier is transparent for all MCloudDB based client applications.

The data is served by multiple backend database servers, which are load balanced by leveraging some dynamic load balancing software, such as Zookeeper [3]. To optimize performance, the load balancers adopt Direct Server Response technique that allows the Backend Data Cloud (BDC) to directly return data to the mobile client applications if necessary. Another key components of MCloudDB are Mobile Data Cloud Infrastructure (MDIC) that comprises of Central MDIC and Regional MDIC. All of MDIC are used for implementing the key value-add data services. Each MDIC is composed of the following key components shown in Figure 2.

- ***Session Management Agent*** – It is responsible for maintaining connections between the MDIC and mobile client applications. Since mobile devices often face the frequent connectivity issue, efficient management of sessions and transactions are required to ensure the seamless connectivity and the ability to restart transactions based on previously saved contexts.
- ***Transaction Management Agent*** – It is responsible for managing and serializing the related database operations required by the mobile client applications.
- ***Multi-tenancy Management Agent*** – It is responsible for collocating customer data depending on customer profile and preference.
- ***Data Caching Agent*** – It is responsible for caching customer data near to user geo-location and/or customer data access point.
- ***BDC Connectivity Agent*** – It is responsible for connecting to the Backend Data Cloud in a cloud agnostic fashion.

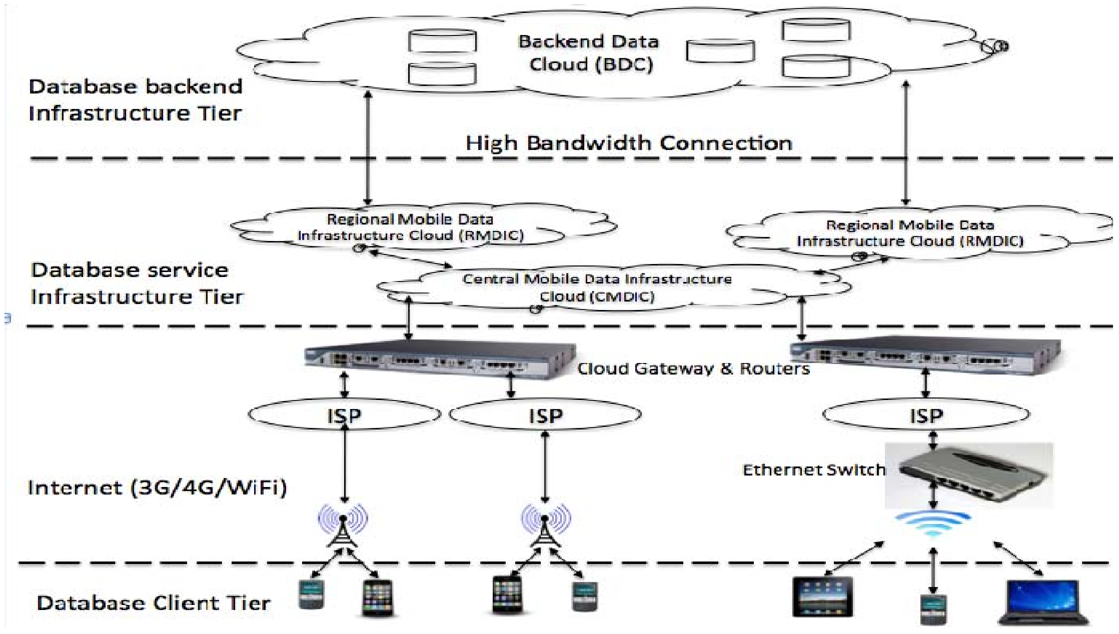

Backend Data Cloud (BDC)
Database backend Infrastructure Tier
High Bandwidth Connection
Regional Mobile Data Infrastructure Cloud (RMDIC)
Regional Mobile Data Infrastructure Cloud (RMDIC)
Database service Infrastructure Tier
Central Mobile Data Infrastructure Cloud (CMDIC)
Cloud Gateway & Routers
ISP
ISP
ISP
Ethernet Switch
Internet (3G/4G/WiFi)
Database Client Tier

Figure 1. The infrastructure of MCloudDB

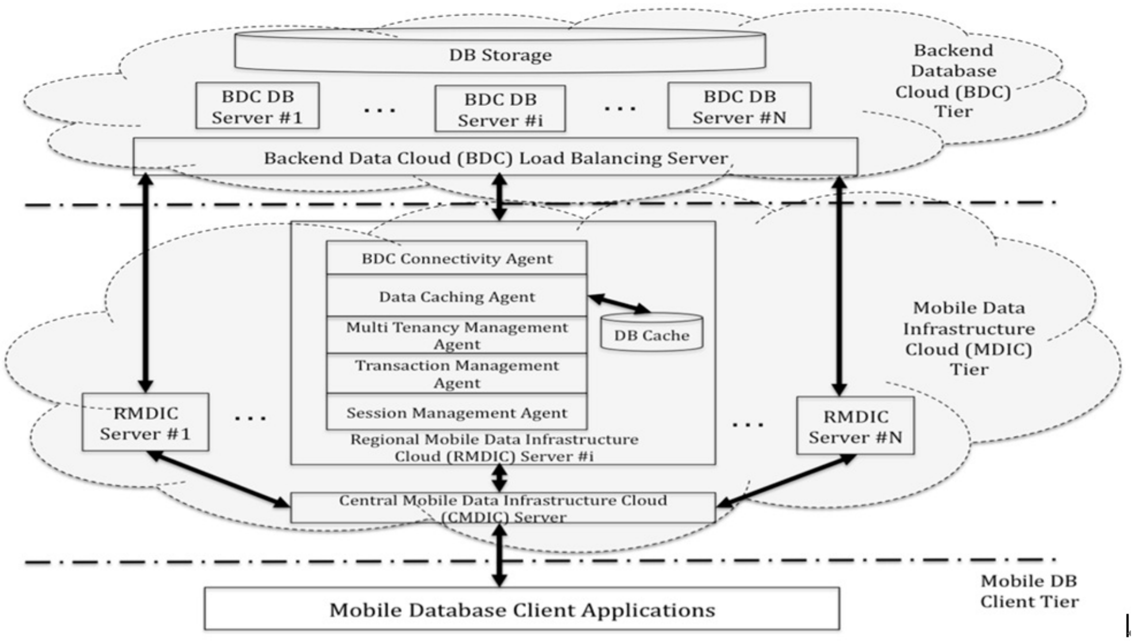

DB Storage
BDC DB Server #1
BDC DB Server #i
BDC DB Server #N
Backend Database Cloud (BDC) Tier
Backend Data Cloud (BDC) Load Balancing Server
BDC Connectivity Agent
Data Caching Agent
DB Cache
Multi Tenancy Management Agent
Transaction Management Agent
Session Management Agent
Regional Mobile Data Infrastructure Cloud (RMDIC) Server #i
Mobile Data Infrastructure Cloud (MDIC) Tier
RMDIC Server #1
RMDIC Server #N
Central Mobile Data Infrastructure Cloud (CMDIC) Server
Mobile Database Client Applications
Mobile DB Client Tier

Figure 2: The key components of the infrastructure

B. *Data Storage Model in MCloudDB*

The data storage model provided by MCloudDB is shown in Figure 3. The data storage model enables data access abstraction (one of the key features in Table 2). First of all, we classify the data into two categories, i.e.,

- Application Data. The data is used and managed by various applications for catering to various user requests. It is expected that typically application vendors would define the schema for this kind of data.
- User Data. The data is private to the user, for example, videos, music, photos, documents etc.

Typically, mobile users wouldn't define the schema for the User Data. Thus, the MCloudDB should define a suitable schema for the User Data. With the MCloudDB, both categories of data are hosted in the multi-tenant databases in BDC, and backed up by the corresponding cache in MDIC.

To abstract the implementation of the backend databases, we introduce a Database Abstraction Layer (DAL) that exposes all types of data management operation (including creation, read, update, and deletion) APIs. DAL really relieves the burden of application developers to make use of vendor specific database constructs. DAL can support multiple databases at the backend, including NoSQL (MongoDB/Cassandra/HBase) and RDBMS

(MySQL at the server and SQLite at the client). Mobile DAL abstracts accesses to the underlying mobile databases. Thereby, Mobile DAL helps mobile applications to be independent of the mobile databases be used.

Depending on the type of data need to be stored, Mobile DAL (or DAL) of MCloudDB chooses the right database and allocates the suitable storage resources for the data. The client applications running in mobile devices cache their critical data in the mobile databases hosted in their local flash devices.

Whenever an application needs to flesh data on its respective server, it would make a call to the Regional MDIC which looks up the requested data in its local cache. A Regional MDIC caches user data depending on the user profile and the latest data accesses. A user can set up its profile to indicate which kind of data they want to cache. The BDC would be accessed if the data can not be found in the RMDIC.

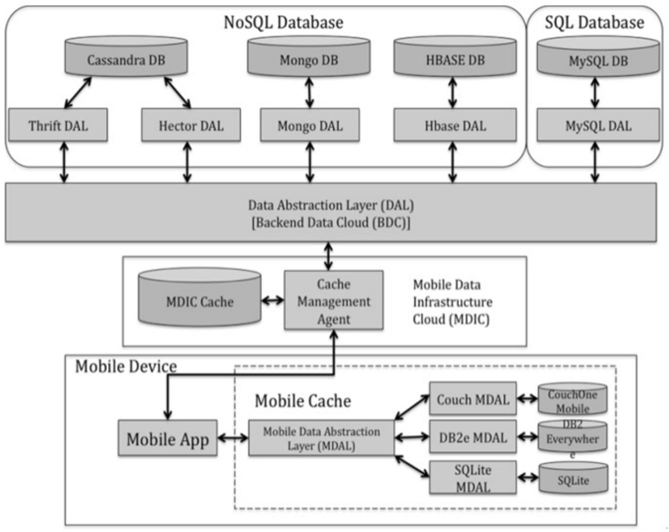


Figure 3. The Data Storage Model for MCloudDB

In short, MCloudDB provides the data access abstraction layer by which mobile applications could use generic database access APIs to access data. So, it can be prevented that vendor lock customers. Moreover, by leveraging MDIC and BDC, mobile applications can interface with the suitable backend databases based on the user specification.

C. *Session Management in MCloudDB*

Session Management and ***Transaction Management*** are two significant components of MCloudDB, as mobile devices may disconnect frequently. Figure 4 provides the illustration of the session and transaction management which enables mobile data transfer resiliency (one of the key features in Table 2).

- Step 1. A mobile application creates its initial connection with the Central MDIC through a getRMDICInstance call.
- Step 2. The Central MDIC maps the nearest Regional MDIC for the client and returns it to the client.
- Step 3. Once the client establishes the connection with the Regional MDIC, it sends an initializeDB request.
- Step 4-5-6-7. The Regional MDIC maps a database access request with the client ID, session ID (**if one exists from previous requests**) and tenant ID by consulting its private metadata tables and creates a transaction handle.
- Step 8. Each client device runs an instance of Mobile Client Development Kit (MCDK) that abstracts the session connection semantics and caches the session handle received from the Regional MDIC.

In a word, MDIC provided by MCloudDB keep track of ongoing transfers which enables resuming unfinished transfers or sessions.

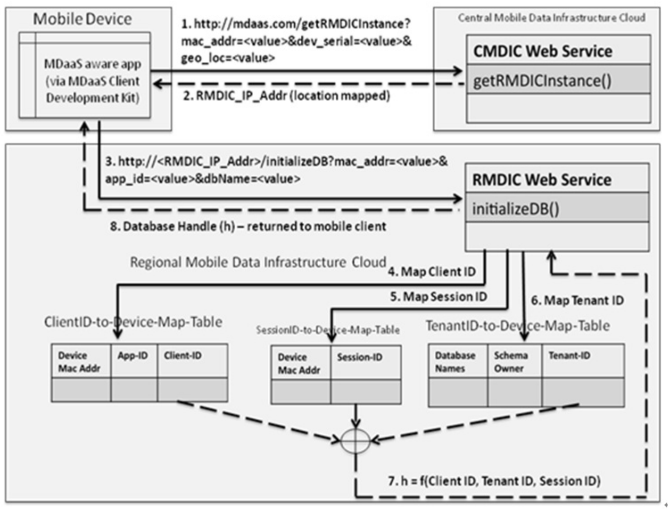

Mobile Device
MDaaS aware app (via MDaaS Client Development Kit)
1. http://mdaas.com/getRMDICInstance?mac_addr=<value>&dev_serial=<value>&geo_loc=<value>
2. RMDIC_IP_Addr (location mapped)
Central Mobile Data Infrastructure Cloud
CMDIC Web Service
getRMDICInstance()
3. http://<RMDIC_IP_Addr>/initializeDB?mac_addr=<value>&app_id=<value>&dbName=<value>
RMDIC Web Service
initializeDB()
8. Database Handle (h) – returned to mobile client
Regional Mobile Data Infrastructure Cloud
4. Map Client ID
5. Map Session ID
6. Map Tenant ID
ClientID-to-Device-Map-Table
SessionID-to-Device-Map-Table
TenantID-to-Device-Map-Table
Device Mac Addr
App-ID
Client-ID
Device Mac Addr
Session-ID
Database Names
Schema Owner
Tenant-ID
7. h = f(Client ID, Tenant ID, Session ID)

Figure 4. The Session and Transaction Management

D. *Multi-tenant based Data Management in MCloudDB*

This section explains the steps of mapping multi-tenant queries illustrated in Figure 5, which enables highly granular multi-tenancy management (in Table 2).

- During initial connecting, the instance of MCDK in the mobile device sends its attributes to the Central MDIC as the attachment with the message getRMDICInstance. The attributes includes Mac address and serial number of the mobile device, as well as the application ID.
- The message getRMDICInstance is dealt with as follows. The Central MDIC contacts the chosen Regional MDIC to convert the mobile device information into a unique handle (by looking up its private tables and encode it), and send the handle back to the application along with the response of the message getRMDICInstance.
- The instance of MCDK in the mobile device saves the handle with the form {tenantID, clientID, sessionID}. If the subsequent queries from the application need to be handled, the MCDK instance would send the handle to the Regional MDIC.
- The Regional MDIC uses the information of the handle to map the query to the right tenant table as shown in Figure 5.

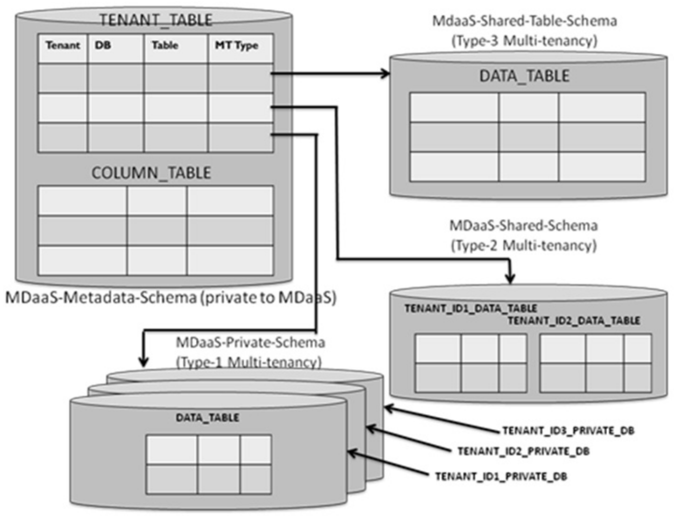


Figure 5. Metadata Tables for Multi-tenancy Management

As illustrated in Figure 5, there are three metadata tables used for the multi-tenancy management.

- ***Tenant Table*** – This table maintains a mapping of the database names, its DB tables, tenant IDs and related types.
- ***Column Table*** – This table maintains a mapping of the tables, the column names with the tenant IDs.
- ***Data Table -*** This table maintains a mapping of the tenant IDs, the column names, and the column value, and contains the actual data.

The requests of the query are processed by the Regional MDIC which firstly maps the requests with the tenant ID (derived from handle as discussed earlier), then retrieves the tenant data from the database in the BDC through the DAL.

In brief, supporting the column-based multi-tenancy scheme enables mobile applications to have multi-tenancy at various levels (i.e., schema-based, table-based, column-based), which not only enhances the utilization of cloud storage but also improves the management of databases.

E. *Data Cache Management in MCloudDB*

This section describes the data caching principles that are applied at various levels to provide seamless connectivity experience to the user, which enables two key features in Table 2 (i.e., mobile data transfer latency reduction, user profile and data filtering).

As indicated in the earlier sections, all recently accessed data are cached at the corresponding Regional MDIC based on user geo-location. For enabling faster access to the User Data, Regional MDIC caches the data based on the user profile. This improves the speed of data access, as at no point of time, the mobile devices fetch data from the actual central (master) database.

In addition, to improve the user experience at the time of connectivity losses, mobile applications can be designed to display the data in their local mobile database caches. The data can be easily accessed through the mobile DAL, when mobile application developers build the mobile applications based on MCloudDB.

Figure 6 describes the data flow of the MCloudDB based mobile applications installed on mobile devices. Nowadays, a well-behaved mobile application often be designed and implemented by making use of Model-Controller-View model. Usually, the mobile database is used as the model; the business logic of the mobile application is utilized as the controller; and the UI of the mobile application is played as the view. The mobile application is able to invoke the APIs exposed by MCDK. An instance of MCDK seamlessly keeps the mobile database (i.e., the model of the mobile application) in sync with the datacenters in Regional MDIC, and eventually with the master database resident in BDC. Besides, the data centers in Regional MDIC are synced periodically with the master databases in BDC.

All in all, the mobile applications are not required to directly interact with the master databases. Instead, they talk to MDIC that abstracts the locality of data for applications, which not only reduces the latency of data transfers but also improves user experience.

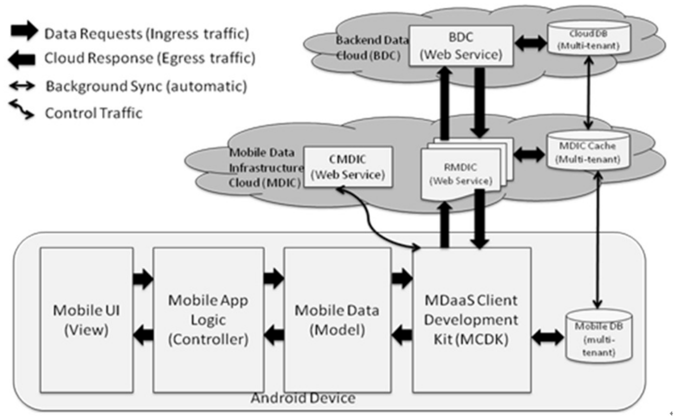

Data Requests (Ingress traffic)
Cloud Response (Egress traffic)
Background Sync (automatic)
Control Traffic
Backend Data Cloud (BDC)
BDC (Web Service)
CloudDB (Multi-tenant)
Mobile Data Infrastructure Cloud (MDIC)
CMDIC (Web Service)
RMDIC (Web Service)
MDIC Cache (Multi-tenant)
Mobile UI (View)
Mobile App Logic (Controller)
Mobile Data (Model)
MDaaS Client Development Kit (MCDK)
Mobile DB (multi-tenant)
Android Device

Figure 6. The Mobile Client Data Model and Path

F. *Scalability Management in MCloudDB*

This section discusses scalability management in Table 2 about how the mobile applications based on MCloudDB scales in a real network wherein there may be millions of mobile users trying to access mobile databases in clouds.

With the Base Station Transceivers (known as Enhanced Node B Controllers – eNodeB in the packet switched 4G world), mobile devices can communicate each other in the IP based Internet core network. Packets originating from the mobile devices and traveling towards the Internet are usually intercepted at the Mobile Gateways which are typically deployed in enterprise routers and then serviced upon.

As shown in Figure 7, a central Mobile Data Infrastructure Cloud (MDIC) and regional MDICs are deployed in the respective mobile gateway routers as dedicated Edge Services. A MDIC communicates with the Backend Data Cloud if the requested mobile data can not be found in the MDIC. On the other hand, if the data can be found in the MDIC, the packets are serviced by the mobile gateway itself. This way reduces the need for the packets to actually reach their Internet destinations.

Data accessing requests are processed by mobile gateways at the packet level to improve mobile data service scalability. Figure 7 shows the deployment infrastructure of MCloudD. Due to limited space, we do not discuss it in details. Table 3 illustrates the different scalability techniques adopted by MCloudDB.

Table3. Scalability Techniques Adopted by MCloudDB

Layer	Technique	Description
Backend Data Cloud	Storage Access Load Balance	Ensures seamless database and storage accesses with scalability.
	BDC service scale-out through new VMs	Load meters spawn new VMs to ensure seamless continuity of service with least latency.
Mobile Data Infrastructure Cloud	Regional MDIC & Central MDIC Service scale-out through new VMs	New VMs are spawned in a regional MDIC that provide the services for Central MDIC requests.
	MDIC Cache replication	As the load on customer data cache increases, the requests are offloaded to the nearby regional MDIC depending on its locality and loads.

To enhance scalability at the access point of MCloudDB based mobile application, Zookeeper [3] load-balancing service is used to streamline coordination between various tasks in the Backend Data Cloud (BDC). In the BDC, Zookeeper maintains a tree of data nodes (called Znode). Depending on a Load Meter, Zookeeper spins up or down additional Virtual Machine instances for accessing the actual databases. The database access nodes (Virtual Machines) are arranged in a hierarchical fashion, like a file-system. Each Znode is coordinated by a master Znode. The mechanisms are described as follows.

- In the BDC, all distributed database servers maintain a copy of the Znode table in their memory.
- A leader server is elected at the beginning.
- All data reading requests are directed to the follower servers, while the master server delegates the updates.
- Update responses are sent back when the servers arrive at a quorum after making the change persistent (on disk).

The software installed on different Regional MDIC servers has a special ZK Client library by which the software knows how to talk to follower servers for reading, or how to send the master server for updating.

In short, MCloudDB based mobile database services enable and support the mobile service scalability (both scale up and scale out) by deploying service instances on different Virtual Machines and using efficient load balancing techniques.

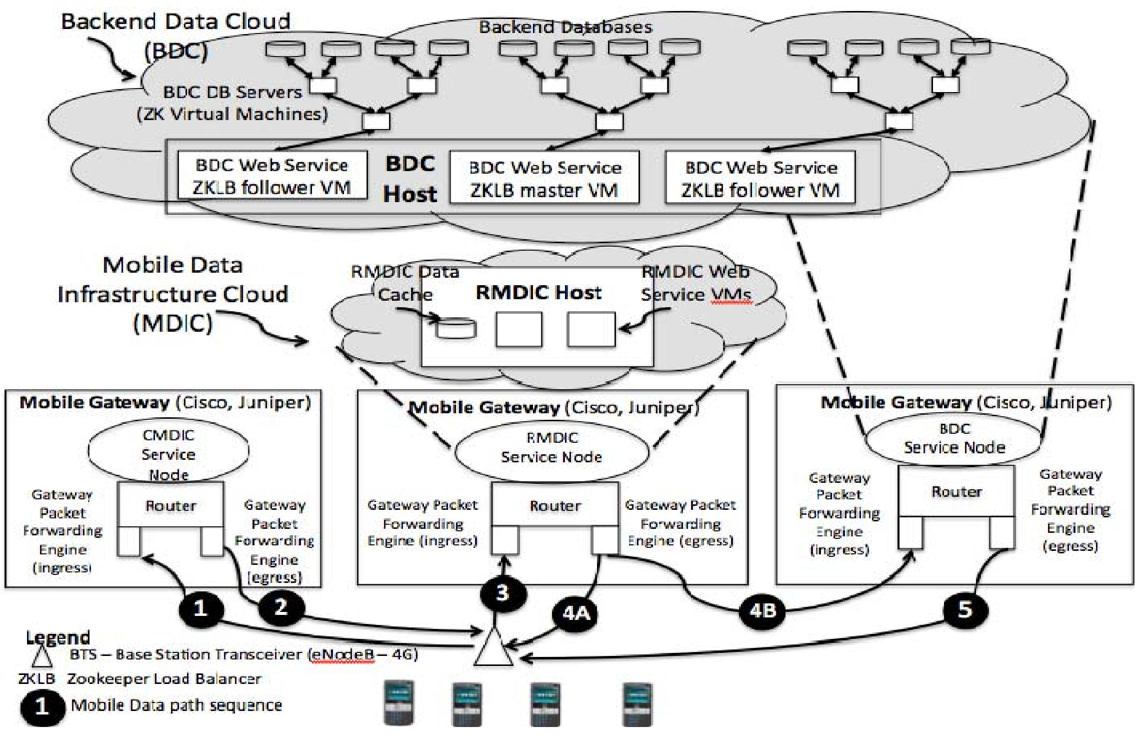

Backend Data Cloud (BDC)
Backend Databases
BDC DB Servers (ZK Virtual Machines)
BDC Web Service ZKLB follower VM
BDC Host
BDC Web Service ZKLB master VM
BDC Web Service ZKLB follower VM
Mobile Data Infrastructure Cloud (MDIC)
RMDIC Data Cache
RMDIC Host
RMDIC Web Service VMs
Mobile Gateway (Cisco, Juniper)
CMDIC Service Node
Router
Gateway Packet Forwarding Engine (ingress)
Gateway Packet Forwarding Engine (egress)
Mobile Gateway (Cisco, Juniper)
RMDIC Service Node
Router
Gateway Packet Forwarding Engine (ingress)
Gateway Packet Forwarding Engine (egress)
Mobile Gateway (Cisco, Juniper)
BDC Service Node
Router
Gateway Packet Forwarding Engine (ingress)
Gateway Packet Forwarding Engine (egress)
1
2
3
4A
4B
5
Legend
BTS – Base Station Transceiver (eNodeB – 4G)
ZKLB Zookeeper Load Balancer
1 Mobile Data path sequence

Figure 7. The MCloudDB Deployment Model

IV. A Case Study

A. *The Case Study Environment*

A prototype application based on MCloudDB has been developed for the validation of MCloudDB. Some of the complex components, such as Database Abstraction Layer, Multi-tenant based Query Management, has been developed and tested by RESTful APIs that are accessible from Android mobile applications. The backend database drivers in the DAL have been developed for most of popular NoSQL cloud databases, such as Cassandra (using Thrift adapter), HBase, MongoDB, and MySQL.

Besides, MCloudDB Client Development Kit (MCDK) has been developed. An instance of MCDK can cache and synchronize the data hosted at the remote cloud based master databases. MCDK has been built using Android's Content Providers that allows mobile applications to access system's SQLite databases using generic APIs. The MCDK is distributed as a separate Java Archive (aka jar) file that can be imported into Android mobile applications so that the Android mobile applications do not require too much programming level intelligence about intricacies of MCloudDB design.

This prototype application was deployed in the Openstack, which is a popular distributed operating system used to build customized and complex cloud infrastructure. One notable advantage of using Openstack, ahead of more popular cloud vendors like Amazon, is that Openstack allows the solution to be deployed on custom hardware and in private data centers. Scaling of control-plane for the MCloudDB based applications is achieved by distributing most processing (relating to subscribers) across multiple virtual machines in the Regional MDIC and the BDC.

Our experiments demonstrate that typical response time for the queries in the MCloudDB based applications depend on the following factors:

- ***The no. of hops for a serviced query***. This number depends on whether the requested data is currently resident in the mobile database (SQLite). The cache in the nearest Regional MDIC cache will be looked up first, and the requested data will be accessed if it is found. If data is still not found, a back-end database (BDC) will be accessed. Meanwhile, the obtained data will be cached for the subsequent accesses.
- ***The No. of fetches for a query***. If a query requires more fetches, then its total response time will be longer.
- Processing power of a rregional mobile data infrastructure cloud (MDIC). This depends on the related CPU power and memory size of automatic allocated VMs.
- The size of the associated database.
- ***The Network bandwidth***.

A graphical representation of network I/O requests and cache updates is illustrated in Figure 8. The Y-axis represents the number of network requests, and the X-axis represents the cache updates for different localities of data.

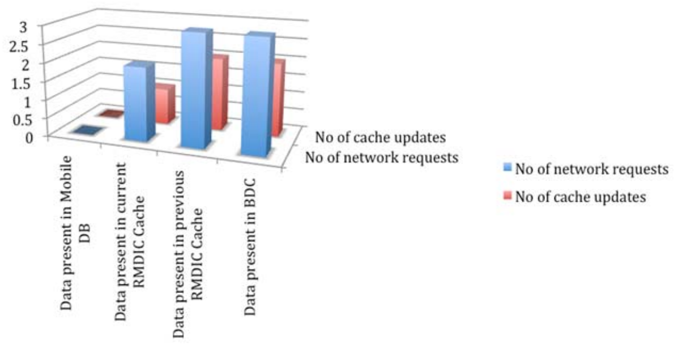


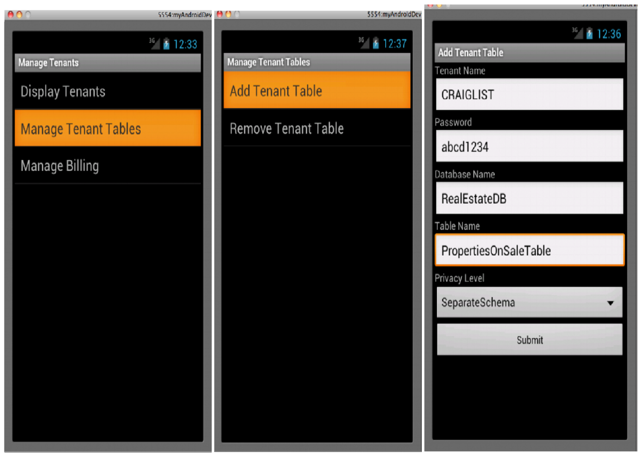
Figure 8. Network Requests and Cache Updates in MCloudDB

B. *Tenant Administration*

The administrative tasks in the MCloudDB based mobile applications can be classified into two main categories, i.e.,

- Administration of the MCloudDB infrastructure. The administration involves provisioning and management of Regional MDIC storage, Regional MDIC and BDC compute node provisioning, image upgrades, automatic load balancing of web services and accesses, etc..
- Administration of multi-tenants data. The administration involves creating, viewing, and altering database schema for tables for each tenant, configuring rules for automatic shard of databases, managing access control lists for users, tenant, meta-data management, and so on.

Besides, various functional components of MDIC support the functionalities in different forms, which are front-ended by the Mobile User Interface administration application. Figure 9 illustrates the user interface for adding a new table called PropertiesOnSaleTable for a tenant Craigslist who has chosen Type-1 Multi-tenancy (Private Schema, Private Table)



Tenant Name: CRAIGLIST
Password: abcd1234
Database Name: RealEstateDB
Table Name: PropertiesOnSaleTable
Privacy Level: SeparateSchema

Figure 9. MCloudDBAdmin App screens for adding tenant tables

C. *Application Data Storage Model Example*

For a MCloudDB based application, it maintains a global metadata table called TENANT_TABLE which stores information about all its tenants. This table is used to vector into multi-tenanted schemea. TENANT_TABLE is maintained in a separate schema called MDaaS-Metadata-Schema in the MDIC. During a schema creation, tenants can specify which tables reside with what level of multi-tenancy. As shown in Figure 10, for Type-1 Multi-tenancy, tenants are provided with their own private schema. For Type-2 Multi-tenancy, tenant tables are collocated within the same schema, but tables are tagged with a Tenant_ID prefix so that data lookups can be faster. Finally, for Type-3 Multi-tenancy, a COLUMN_TABLE (metadata) is maintained by MDaaS, which holds which column mapping and actual data is stored in DATA_TABLE.

Figure 10 shows the data storage model of the prototype application. It demonstrates how the data of multiple tenants are collocated in the same database and within the same table. The type of a tenant should be given when the tenant creates the debatable. For like-minded tenants (e.g., Realtor INC and Taxman ORG) who choose the same category (e.g.,Type-3 Multi-tenancy), their data are collocated and they share the same table (e.g., PropertyInfoTable is created by Realtor INC and is shared by Taxman ORG).

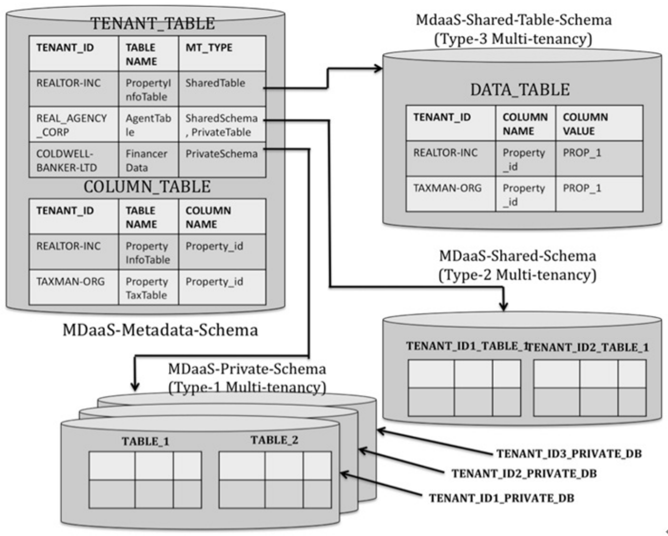


Figure 10. The data storage model of the prototype application

V. Conclusion and Future Work

This paper proposes a novel mobile cloud database service solution to efficiently store and manage mobile data on cloud databases for mobile applications. This solution not only provides a common framework which plays as an efficient mobile data access channel between mobile applications and cloud databases, but also overcome the limitations of existing mobile databases by providing some desirable mobile cloud computing features, such as multi-tenancy and elasticity. Meanwhile, it still offers mobile applications with major mobile database features, such as mobility and location-aware, automatic transaction failover for aborted sessions. In short, this research work provides a new direction to provide mobile cloud databases for mobile applications on clouds in the future. Based on the experiment results from our prototype application, we believe that the proposed MCloudDB approach provides an improved solution supporting mobile data management and accesses for mobile applications by leveraging the current cloud databases.

Our future research work includes two areas. One of them is focusing on mobile cloud database security and stability issues and solutions. The other focusing area is location-based mobile data accesses processing and load balance to reduce communications and energy.

References

[1] RoselinSelvarani, D., and T. N. Ravi. "A survey on data and transaction management in mobile databases." International Journal of Database Management Systems (IJDMS), Vol.4, No.5, October 2012.

[2] Hassan Artail, HaidarSafa, RanaELZinnar, and HichamHamze. "A Distributed Database Framework from Mobile Databases in MANETs," Proc. of Third IEEE International Conference in Wireless and Mobile Computing, Networking and Communications (WiMOB), 2007.

[3] Serrano-Alvarado, Patricia, Claudia Roncancio, and Michel Adiba. "A survey of mobile transactions," Distributed and Parallel databases, Vol. 16, No. 2, pp: 193-230, 2004.

[4] SumitiSehgal, Deepti Arora, and ShabnamKwatra, "Review of Issues in Location Aware Query Processing in Mobile Databases," The International Research Journal of Social Science & Management, Vol.1, No.12, 2012.

[5] DikLun Lee, Jianliang Xu, Baihua Zheng, and Wang-Chien Lee. "Data management in location-dependent information services". Proceedings of IEEE 20th International Conference on Pervasive Computing and Data Engineering, 2004.

[6] Vijay Kumar. "Mobile Database Systems", Wiley-Interscience, John Wiley & Sons, Inc. Publication, New Jersey, 2006.

[7] Salman Abdul Moiz and Lakshmi Rajamani, "Replication Strategies in Mobile Environments," International Journal of Information Technology, Vol 2, No.1, 2009.

[8] Ajay P.Chendke, and Mrs SS Sherekar, "Comparative Study of Query Processing Architectures in Mobile Environment," International Journal Of Computer Science And Applications, Vol.6, No. 2, 2013.

[9] A. Ahmed, D. D. Dominic, and A. Abdullah. "A Novel Replication Strategy for Large-Scale Mobile Distributed Database Systems," Science and Technology, Vol. 6, No. 3, 2011.

[10] Can Türker and Gabriele Zini, "A Survey of Academic and Commercial Approaches to Transaction Support in Mobile Computing Environments", Technical Report. Department of Computer Science, Institute of Information Systems, 2003.

[11] Sedivy, Jan, Tomas Barina, Ion Morozan, and AndreeaSandu. "MCSync-Distributed, Decentralized Database for Mobile Devices," Proc. of IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2012.

[12] Fageeri, Sallam, and Rohiza Ahmad. "Mobile Data availability Similarity between Cooperative Caching and Replication," IOSR Journal of Computer Engineering , Vol1, No. 4, 2012.

[13] Tsai, Wei-Tek, Yu Huang, Xiaoying Bai, and Jerry Gao. "Scalable Architectures for SaaS." Proceedings of Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012.

[14] Ruay-Shiung-Chang, Jerry Gao, Volker Gruhn, Jingsha He, George Roussos, and Wei-Tek Tsai, "Mobile Cloud Computing Research - Issues, Challenges and Needs", Proceedings of IEEE Seventh International Symposium on Service-Oriented System Engineering, 2013.

[15] Elmore, Aaron J., Sudipto Das, Divyakant Agrawal, and A. E. Abbadi. "Towards an elastic and autonomic multitenant database."In Proc. of NetDB Workshop. 2011.

[16] Dey, Akon, Alan Fekete, and Uwe Röhm. "Scalable transactions across heterogeneous NoSQL key-value data stores." Proceedings of the VLDB Endowment, 2013.

[17] Shuyu Li, Jerry Gao. "Mobile Data Services – Moving from Mobile Databases to Mobile Cloud Data Services". Submitted for publication