

目錄

1 DMSync概述	3
1.1 前言	3
1.2 DMSync 介紹	3
1.3 DMSync結構	4
1.4 DMSync元件	5
1.5 DMSync工作過程	8
1.6 DMSync CD 包裝清單	9
1.7 DMSync典型應用	10
1.8 DMSync支援的平臺	11
1.9 DMSync 同步選項	11
2DMSync Server環境	12
2.1 DMSync DB創建	12
2.2 DMSync系統表的創建	12
2.3 Jetty 構建	15
2.4 Jetty DMSync.XML配置	16
2.5 Jetty Server啓動	17
3DMSync GLOG環境	18
3.1 GLOG概述	18
3.2 DMSync MDB GLOG環境	19
3.3 DMSync RDB GLOG環境	26
4部署DMSync 同步	32
4.1 增加DMSYNC USER	32
4.2 註冊遠端資料庫RDB	32
4.3 增加PUBLICATION	33
4.4 增加SUBSCRIPTION	34
4.5 增加USER COMMAND和OBJECTION	34

4.6	增加RDB Subscription訂閱-----	38
4.7	RDB的其他配置-----	39
5	使用DMSync同步-----	40
5.1	數據準備-----	40
5.2	DMSync進行同步-----	40
6	DMSync同步出錯處理-----	42
7	同步資料衝突-----	43
8	不同類型的映射-----	45
9	DMSync性能調優-----	47
9.1	PackageSize-----	47
10	----- DMSync演示	
	48	
10.1	運行環境-----	48
10.2	初始化DMSync DB-----	48
10.3	初始化MDB-----	49
10.4	啓動Jetty-----	49
10.5	在Android上初始化RDB并執行同步-----	50
10.6	MDB資料更改的同步-----	54

1 DMSync 概述

1.1 前言

- 本手冊假定您已經瞭解 Java Servlet 及 JDBC 的相關內容。
- 本手冊假定您已經瞭解 DBMaker 的使用或具有 RDBMS 使用經驗。
- 本手冊假定您已經對 HTTP Web Service 有一定的瞭解。
- 本手冊所有 DMSync 的功能描述都基於 DMSync1.0。

1.2 DMSync 介紹

DMSync 是一套同步技術，用於協助資料庫之間的資料交換，它可以解決不同位置、不同類型的應用程式之間的資料共用問題。在基本的 DMSync 應用中，可以簡單地將資料庫分為 Master DB (MDB) 和 Remote DB (RDB) 兩種。這裏的 MasterDB 是指運行在資料 Server 上且支援 JDBC 的關係型數據庫，是主要的資料存儲倉庫。用戶可以通過合理地使用 DMSync，將一個或多個 RDB 的資料進行有效、一致地存儲，如 DBMaker 等各種關係型數據庫。而 RemoteDB 則是指那些運行在存儲能力和運算能力比較有限的平臺上的資料庫，此類資料庫不宜做大量、持久性的資料存儲，如 Android 移動手持設備的 SQLite。

DMSync 作為一套成熟的資料同步解決方案，主要有以下特點：

基於 WebService 的同步

DMSync 可以理解成一種網路服務。當 MDB 或 RDB 準備進行同步時，用戶只需要通過網路來訪問 DMSync WebService 就可以進行快捷方便的同步，而且 DMSync WebService 採用的是 SOAP 協定標準。

支援多種類型資料庫

通常情況下，DMSync MDB 是指支援 JDBC 的關係型數據庫，即任一種支援 JDBC 的資料庫產品都可以作為 DMSync 的 MDB 來運行，這樣可集中存儲各種來源的資料，如支援 JDBC 的 MSSQLSERVER, MYSQL 等。對於 DBMaker 而言，現階段僅支持的版本為 DBMaker5.2。

RDB DMSync 則必須要求是 SQLite，SYSCOM 為 Android SQLite 開發了專有的工具 **SYSCOM JDBC Driver for SQLite on Android**，用戶也可以使用 SYSCOM JDBC Driver 來操作 SQLite。

支持多個 MDB 和 RDB

在設置 MDB 和 RDB 間的資料同步規則時，用戶可以簡單地將一個或多個 RDB 資料，有選擇性地同步至一個或多個 MDB 中，請參見表 1.1：

RDB	MDB
1	1
1	N
N	1
N	N

N:指代兩個或兩個以上

表 1.1

多個同步實現方法

若將資料庫間的資料進行同步更新並保持完全一致，可採用多種方法。DMSync 支援 **GLOG** 和 **TIMESTAMP** 兩種方式。

GLOG 方式——記錄資料庫中資料表變更的前映射值和後映射值稱為 LOG 記錄，記錄資料庫中所有發生資料變更的表稱為 GLOG 序列。DMSync 依據 GLOG 序列來確定變更的資料表，再依據資料表的 LOG 記錄來確定對應的 DML 操作和前後映射值，接下來根據用戶設置的規則，通過 WebService 將這些資料的更改一併發送到另一端資料庫，並最終應用資料更改。

TIMESTAMP 方式——首先給資料表添加 timestamp 類型的欄位，這種方式的同步是以時間為依據的，每次執行同步時，會將上次同步時間以後的所有資料更改進行同步。

完整的資料衝突解決策略

在兩個大併發量的資料庫間進行同步時，所產生的記錄往往會具有相同的主鍵。DMSync 內置了完整的衝突解決策略，即 MDB 為主和 Modify Time。DMSync 將主鍵作為解決衝突的依據，因此用戶在使用 DMSync 時應儘量避免使用 UPDATE 來更改主鍵。

支援不同類型的表字段同步

DMSync 進行資料同步時，會使用同步規則中用戶編寫的 UserCommand。UserCommand 主要包含 MDB 表字段與 RDB 表字段的對應映射關係，DMSync 對表的 schema 及欄位類型不敏感，可以採用不同的表名稱、欄位名稱或欄位類型，通過適當的設置進行資料同步，但同時也應該注意到各種類型之間的相容性。有關這方面的介紹，DMSync 高級-類型映射章節會作詳細地介紹。

1.3 DMSync 結構

DMSync 可以簡單地分成三部分：DMSyncServer, MDB, RDB。如圖 1.1。

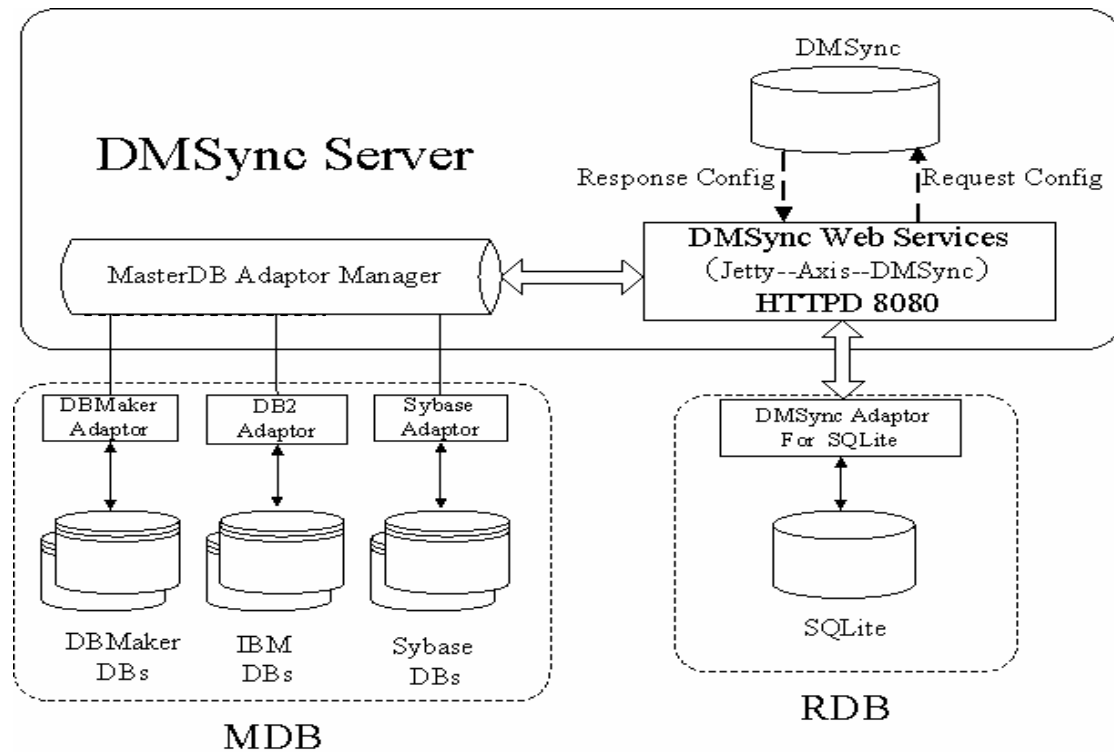


圖 1.1

DMSync Server

DMSync Server 包含一系列集成元件，可以完成 RDB 和 MDB 之間的同步。

DMSync Server 的主要組件有：

- DMSync DB：一個存儲重要配置資訊的資料庫，如 publication 等。
- Jetty：一個 Java Servlet，用來提供 http Webservice。
- Axis2：以 SOAP 協定標準封裝傳輸的資料。

MDB

嚴格來講，MDB 指包含 DMSync 支援的資料庫伺服器，這裏的 DMSync 支援指 DMSync 提供的對該資料庫的支援程式。一般來講，MDB 應該是穩定的關係型數據庫，如 DBMaker。MDB 既可以與 DMSync Server 運行在同一主機上，也可以部署在離 DMSync Server 較近的獨立主機上。

RDB

遠端資料庫節點運行在遠端設備上，RDB 中不僅要存儲資料，還要存儲 DMSync 的相關配置資訊，如 Subscription。既相對於 DMSync Server，RDB 也要作為 DMSync Client 向 DMSync Server 發起同步請求。

在大多數的實際應用中，往往會將多個 RDB 中的資料同步到一個 MDB 中，因此在 DMSync 資料庫中，需要存儲所有已註冊的 RDB 的相關資訊。需要說明的是多個 RDB 可以是多個不同設備上的資料庫，也可以是同一設備上的多個不同的資料庫。

1.4 DMSync 元件

在介紹 Dmsync 工作過程之前，我們先為大家介紹 DMSync 中的幾個重要概念。

Publication

Publication 可以理解成發佈，簡稱 PUB。相對於 DMSync Client 來講，意味著已經確定了同步規則、同步方式和同步 MDB 位置。一個 DMSync Server 上的所有 Publication 資訊都存儲在表 DMSYNC_SYNC_PUB 中，請參考 DMSync Server 環境構建章節以獲取更詳細的資訊。如下圖 1.2 所示：

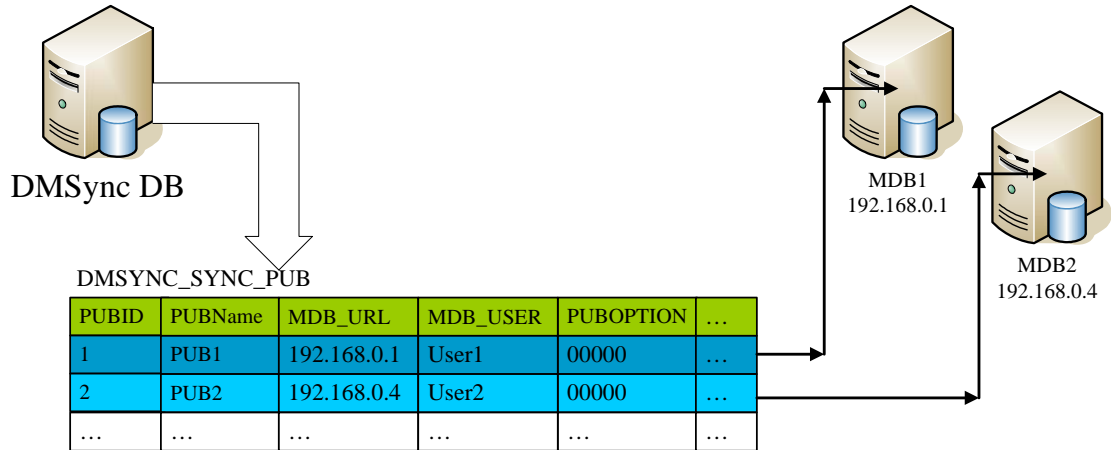


圖 1.2

Objection

Publication 只給出了同步資料表所在的資料庫及位置，而 Objection 則給出了具體的同步關係和規則，包括 MDB 和 RDB 的同步資料表名稱以及兩個表的主鍵定義欄位，衝突解決策略確定了兩個表字段映射關係的 UserCommand。如下圖 1.3 所示。

從圖中我們可以看出，每個 Objection 通過 PUBID 關聯一個 Publication。Objection 可以簡稱為 OBJ，所有的 Objection 資訊都將存儲在表 DMSync DB DMSYNC_SYNC_OBJ 中。

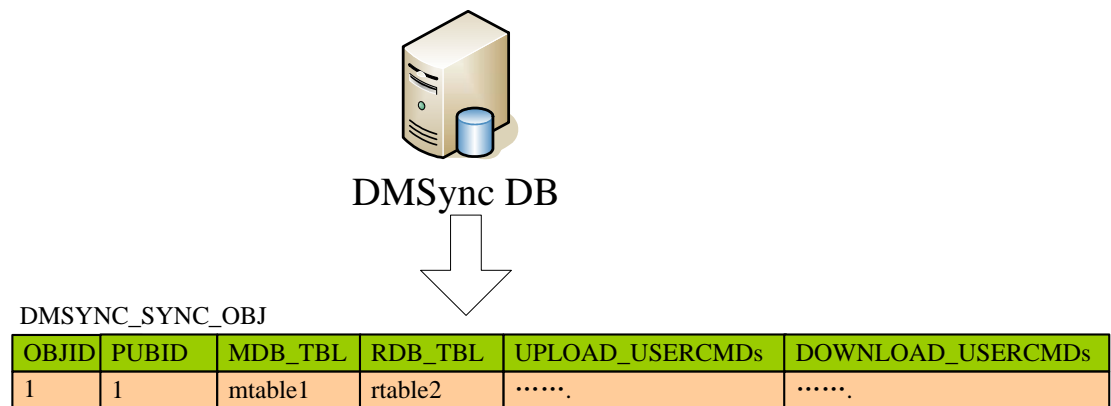


圖 1.3

Subscription

Subscription 在 DMSync 中理解為訂閱，代表執行一次 DMSync 同步的執行許可。Subscription 簡稱為 SUB，包含了可以執行同步的 RDB、使用的 Publication、最後成功執行的時間以及記錄數位移 OFFSET。SUB 資訊存儲在 DMSync 資料庫的 DMSYNC_SYNC_SUB 表中，如下圖 1.4 所示，從圖中我們可以看出 SUB 存儲了關聯資訊。

DMSYNC_SYNC_SUB

RID	SUBID	USERID	PUBID	OFFSET	...
1	1	1	1	20	...
...

圖 1.4

UserCommand

UserCommand 解釋為可執行的用戶命令，是 DMSync 確定表字段的映射關係。UserCommand 中的一些運算式稱為 NamedParameter，最常用的是{\$NEW}和{\$OLD}，分別代表另一端資料庫的前、後映射值。DMSync 要求為兩端資料庫中資料表的 IUD 操作編寫對應的 UserCommand，這樣才能完整地進行同步。IUD UserCommand 和標準的 IUD SQL 語句十分類似，區別在於 VALUES 部分，IUD UserCommand 中的所有 VALUES 要以 \$NEW 和 \$OLD 運算式開頭並和對應表的欄位組合在一起，表示從另一端資料庫的表中取值。

UserCommand 中的主要內容就是確定要進行同步欄位之間的映射關係，在執行同步時，這些命令將被執行。UserCommand 不會獨立存在，而是作為 Objection 的一部分，詳細內容將在後面的章節進行講述。

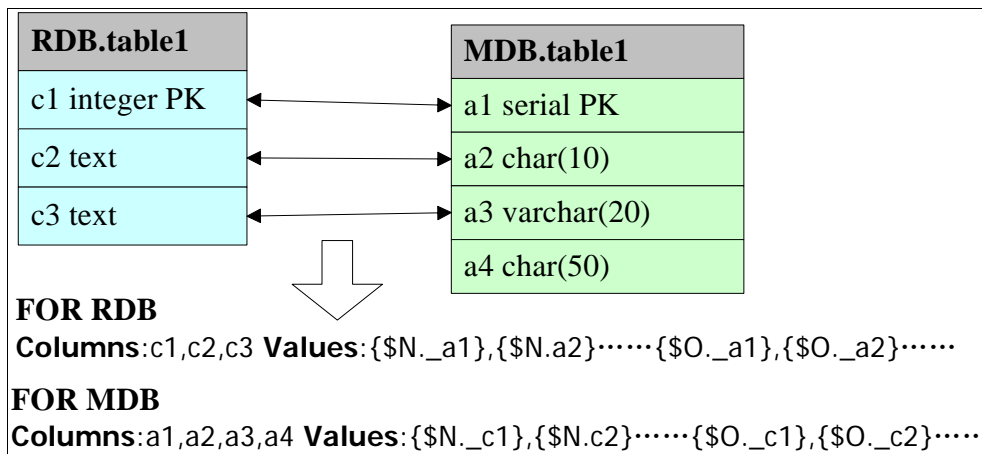


圖 1.5

OFFSET

OFFSET 指成功的位移。每次同步成功時，DMSync 會統計同步成功的筆數並將其記錄到具體的 Subscription 中。

RemoteSubscription

RemoteSubscription 簡稱為 RSUB，從名稱上就可以判斷出 RSUB 和 Subscription 是對應關係，且存儲在 RDB 上。RSUB 就是某個 RDB 訂閱成功的具體資訊，主要指明了默認 DMSyncServer 的位址和同步時使用的默認用戶名。

UserName

這裏的 UserName 實際上是同步時連接 DMSyncServer 所需的認證用戶名，在 DMSync 資料庫表的 DMSYNC_SYNC_USER 中註冊了所有用戶。RDB 要同步時必須指定用戶名，UserName 必須在同步請求的連接串 ConnectStr 中指定，如果 ConnectStr 中沒有指定，那麼同步時會使用 RSUB 中的默認用戶名。

ConnectStr

ConnectStr 指連接串，主要包含 DMSync Server 位址和同步的 PUBNAME，如下圖 1.6 所示。ConnectStr 和 RSUB 有直接的關係，ConnectStr 中沒有表現的參數，DMSync 會在必要的時候檢索 RSUB 中的必要資訊。ConnectStr 是作為 DMSync Client API 的參數來使用的，即在程式中使用。

ClientConnectString

```
dmsync://192.168.0.10:Port/connect?pub=pub1&username=user1&version=...
```

圖 1.6

PubOption

PubOption 是 Publication 的一部分，由 5 個數位組成的字串，預設值為 00000，其中每一位元數字都有具體的含義。

00000

1 st Value: 交易類型 0: transaction, 1: resume

Transaction: 事務類型 after upload server commit, after download client commit

Resume: 每一筆 record 的操作 (insert / update / delete), 進行 commit 一次

2 nd Value: 上傳下載 0: both, 1: upload, 2: download 標示此 publication 同步時資料處理

3 rd Value: 同步規則 0: glog sequence, 1: timestamp 目前僅實現 glog 方式

1.5 DMSync 工作過程

一般情況下，DMSync Server 會長時間穩定地以 WebService 的方式提供服務，RDB 的應用程式要通過調用 DMSync Client API 來完成同步。DMSync 提供了 4 個 API 介面: init(), connect(), sync(), disconnect(), 用戶需要嚴格按順序來執行這 4 個介面才能使用 DMSync。運行 init() 時會檢查各種配置資訊是否正確，運行 Connect() 時會根據用戶提供的 ConnectStr 連接至指定的 DMSyncServer；運行 Sync() 時會使用 ConnectStr 中的 PUBNAME 檢索出 DMSync 的相關配置資訊，並根據欄位的映射關係來完成資料同步；最後使用 disconnect() 斷開連接，釋放使用的 DMSync 的資源。在使用 DMSync 時，用戶應嚴格按順序調用這些 API。如圖 1.7

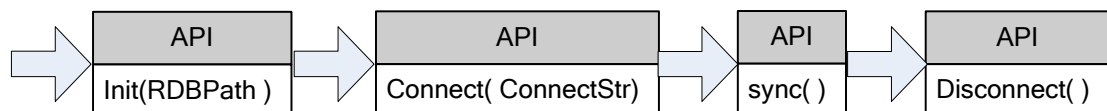


圖 1.7

Init (RDBPath)

Init(String RDBPath) 表示用戶需要傳入一個參數，即 RDB 路徑。因為 RDB 不僅相對於 MDB 可作為遠端資料庫來存儲資料，還可相對於 DMSync Server 作為 DMSync Client 來使用，因此 RDB 中存儲了一些 DMSync Client 的配置資訊。當用戶調用 init() 時，會檢查這些配置資訊，包括是否存在表 RDB_SYNC_SYS，RDB_SYNC_RSUB 等，同時 init() 還會檢查資料表是否擁有主鍵，DMSync 要求每一張資料表都必須擁有主鍵。

在使用 RDBPath 時，請注意使用絕對路徑。

Connect (ConnectStr)

ConnectStr 是 Connect () 的必要參數，ConnectStr 指定了 DMSync Server 的地址，還包括 PubName。當 Connect() 連接至 DMSync Server 後，會下載與 PubName 相關的資訊，包括 Objection。此時，DMSync 又會根據下載的 Objection 來檢查資料表的資訊。

Sync ()

當執行 Sync () 後，DMSync 就要開始進行資料同步了。在執行同步時會根據下載的 Objection 資訊並按照規則將 Client 端的資料更改封裝成 xml 資料包，然後通過 DMSync WebService 將這些資料包發送至 MDB 端進行應用更改。與之類似的是在執行 Sync () 時，MDB 端的資料更改也會通過 WebService 發送至 RDB 端。

當執行 Sync() 時，如果兩端資料庫都進行了更改，DMSync 會首先 UPLOAD RDB 的資料更改，然後再 DOWNLOAD MDB 的資料更改。這兩個過程可以通過設置 PUBOPTION 來進行控制。

Disconnect()

執行完 sync() 後，用戶可調用 disconnect() 函數以釋放佔用的資源。

1.6 DMSync CD 包裝清單

DMSync CD 包含 DMSync Server 端和 Client 端的套裝軟體。Server 端套裝軟體是一組用來運行伺服器的套件，Client 端套裝軟體則提供 RDB 端應用程式需要使用的 Java Reference Library。

Server Package

Server 端相關元件全部部署在 Jetty&&Axis2 套件下，Jetty 是一個 Java Servlet 容器，axis2 是一個 SOAP 協定元件。

將 Server Package 解壓後，目錄結構如下圖 1.8 所示。

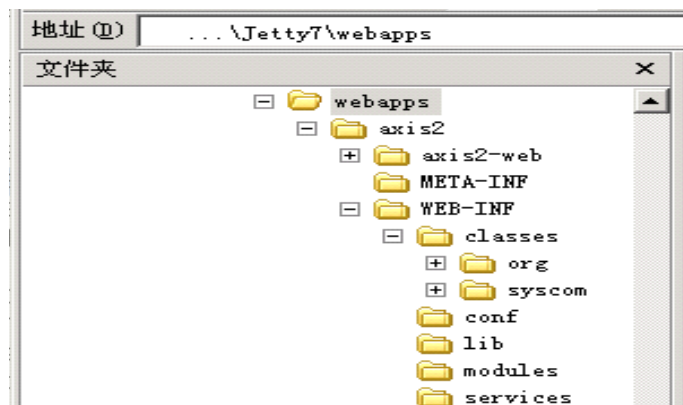


圖 1.8

Server Package 由許多獨立模組構成，這些模組有：

1) **dmsync-x.x.aar**：DMSync service 包是 DMSync 的主要組件，xx 代表版本號，由具體數位替代。dmsync-x.x.aar 位於上圖所示的... \Jetty7\webapps\axis2\WEB-INF\services 目錄下。當 DMSync 需要升級時，只需將新版本複製到對應的檔夾中即可完成發佈。

2) **syscom class** 目錄：DMSync timestamp 1.1 已經移除，故此處不做解釋。

3) **dmsync.xml**：DMSync Server 配置檔，可以配置包括 DMSync DB 位置在內的多項內容，位於... \Jetty7\webapps\axis2\WEB-INF\conf 目錄下。

4) axis2.xml : axis2 的配置檔，位於...\\Jetty7\\webapps\\axis2\\WEB-INF\\conf 目錄下。

5) dmjdbc30.jar : DBMaker JDBC Driver，位於\\Jetty7\\webapps\\axis2\\WEB-INF\\lib\\目錄下，前面章節已經做過介紹，要求 MDB 是必須支持 JDBC 的 RDBMS，而 dmjdbc30.jar 則是 DMSync 用來支持 DBMaker 的，但並不是將任意 RDBMS 的 jdbc driver 放入 lib 檔夾中，都可以將其作為 MDB。有關 MDB 所支援的產品列表請參考 DMSync Release Notes。

6) ehcache-1.4.0.jar 等其餘 jar 檔存放在 lib 檔夾中，是 jetty 和 axis2 用來實現緩存功能的，DMSync 並不要求用戶為此作配置修改。如果需要更詳細的介紹，請參考各個 library 的官方網站。

7) DMSync DB SQL Script : 用來初始化 DMSync 資料庫的腳本。

另外，Jetty和Axis2 都是開源產品，用戶可以參考<http://www.eclipse.org/jetty/>和<http://ws.apache.org/axis2/>下載內容。還可以使用DMSync發佈已經配置好的版本，或下載官方版本配置。DMSync使用的版本是jetty - hightide -7.0.1. v20091125。

Client Package

DMSync Client 提供的內容比較簡單，只有一個 DMSyncAndroidClient1.01.jar 和一個 RDB SQL 腳本。前者用來支援 Android 平臺上的 SQLite 以作為 DMSync RDB，同時也為之提供了 SYSCOM JDBC Driver，後者則是要求 RDB 創建 DMSync RDB 系統表的參考腳本。對於 DMSyncAndroidClient1.01.jar 來說，其中的數字代表版本號，有 Java 開發經驗的人將其作為 Referenced Library 也是十分簡單的。包結構如下圖 1.9 所示。



圖 1.9

DMSyncAndroidClient1.01.jar是為android 平臺準備的Referenced Library，有關如何搭建Android SDK開發平臺請參考www.android.com。

1.7 DMSync 典型應用

C/S 模式應用程式

在實際應用中，有時需要將多個終端資料傳送至一個集中的資料庫中。部署 DMSync

後，可以輕易將多個 Client 端資料同時傳送至資料 Server 端，另外還可以通過對 DMSync 的參數調整來支援高並行、大資料量的同步。

移動和無線應用程式

當前的移動業務和移動辦公已經十分普及，DMSync 會通過 WIFI 或 EDEG 為移動業務進行即時的資料處理。既可以從公司或中央站獲取緊急資訊，也可以將終端處理的重要資料回饋至中央資訊庫。

1.8 DMSync 支援的平臺

DMSync Server 支援的平臺

DMSync 目前僅能夠部署在 Window 平臺上。

MDB 支援的平臺

MasterDB 目前僅指運行在 Windows 平臺上的關係型數據庫，如 DBMaker。

RDB 支援的平臺

RemoteDB 目前僅支持 Windows 或 Android 平臺上的 SQLite。

1.9 DMSync 同步選項

實際應用中的資料操作和同步，應用程式往往只需要完成一部分功能即可，如僅需上傳。DMSync 可以通過選項設定以達到同樣的效果。

UPLOAD

表示從 RDB 同步資料至 MDB，但不下載 MDB 中存放的其他用戶更改。

DOWNLOAD

從 MDB 下載所有資料更改，忽略本用戶對資料所做的更改，相當於唯讀。

BOTH

包括 UPLOAD 和 DOWNLOAD 兩個過程，順序是先執行 UPLOAD 再執行 DOWNLOAD。

2 DMSync Server 環境

上一章已經對 DMSync 的基本知識作了介紹，其中提到的 DMSync Server 主要包括 Jetty&&Axis2 和 DMSync DB。前者用來提供 WebService 和支援 SOAP 協定標準，後者用來存儲 DMSync 的一些配置資訊。它們可以運行在同一台主機上，也可以運行在兩台主機上，本手冊中所指的是運行在同一台主機的情況。DMSync CD 中已經提供了部署完整 DMSync DB 所需要的檔和相關 Schema。

本章僅為幫助用戶加深理解 DMSync、解釋 DMSync 資料庫參考腳本、建議用戶使用 DMSync 安裝時的預設環境。

Jetty 啓動時會從 DMSync 資料庫中讀取一些配置資訊，因此會先介紹 DMSync 資料庫的參考腳本。

DMSync.sql 中主要是創建 DMSync 資料庫和系統表的 SQL 命令，用戶可以使用 dmSQL 命令行工具來執行腳本。這些系統表如下表 2.1 所示。

系統表名稱	注釋
DMSYNC_SYNC_OBJ	存儲 MDB table 和 RDB table columns 的同步映射關係
DMSYNC_SYNC_PUB	存儲 publication
DMSYNC_SYNC_RDB	存儲註冊的遠端資料庫
DMSYNC_SYNC_SUB	存儲 subscription
DMSYNC_SYNC_USER	存儲 DMSync Server 的用戶

表 2.1

2.1 DMSync DB 創建

DMSync DB 用來存儲 DMSync 的重要資訊，因為 DMSync DB 使用的是 DBMaker，因此需要在 DBMaker 中創建 DMSync 資料庫。

```
dmSQL>CREATE DB DMSync;
```

注意在 dmconfig.ini 中填寫相對的 IP 位址和埠號

當資料庫 DMSync 創建成功後，可接著創建表 2.1 中列出的系統表。另外 DMSync 還要求 DMSync DB 和 MDB 的 LCODE 必須保持一致，即在確定了 DMSync DB 的 LCODE 之後，創建 MDB 時還必須採用同樣的 LCODE。DBMaker 的 LCODE 可以在配置檔 dmconfig.ini 中進行配置，詳情請參考 DBMaker 資料庫管理員手冊。

2.2 DMSync 系統表的創建

這些系統表都存儲在 DMSync DB 中，本節將詳細介紹這些系統表的 schema。

DMSYNC_SYNC_PUB

用來存儲作為同步依據的 publication，指明了同步資料庫 MDB 的位置和同步選項。在用戶使用 DMSync 時，通過指定 publication name 來確定一個同步，並通過 publication name 來確定 MDB 的位置和對應的 Objection 以及 Subscription 資訊，以此來進行同步。

DMSYNC_SYNC_PUB

欄位	類型	注釋
PUBID SERIAL(1)	PRIMARY KEY	publication 的 ID
PUBNAME	VARCHAR(32)	publication 的名稱。
MDB_URL	VARCHAR(128)	此 publication MDB 的地址，地址是 JDBC URL。
MDB_USERNAME	VARCHAR(32)	DMSync 訪問 MDB 資料庫使用的用戶名
MDB_PASSWORD	CHAR(32)	DMSync 訪問 MDB 時使用的密碼，明文形式。
PUBOPTION	VARCHAR(32)	PUBOPTION 選項

PUBOPTION : 00000

1 st Value:交易類型 0: transaction, 1: resume

Transaction: 事務類型 after upload server commit, after download client commit

Resume: 每一筆 record 的操作 (insert / update / delete), 進行 commit 一次

2 nd Value:上傳下載 0: both, 1:upload, 2:download 標示此 publication 同步時資料處理

3 rd Value:同步規則 0:glog sequence, 1: timestamp 目前僅實現 glog 方式

SQL 參考腳本:

```
CREATE TABLE SYSADM.DMSYNC_SYNC_PUB (
    PUBID SERIAL(1) PRIMARY KEY,
    PUBNAME VARCHAR(32) NOT NULL ,
    MDB_URL VARCHAR(128) NOT NULL ,
    MDB_USERNAME VARCHAR(32) NOT NULL ,
    MDB_PASSWORD CHAR(32) NOT NULL ,
    PUBOPTION VARCHAR(32) DEFAULT NULL
);
```

DMSYNC_SYNC_OBJ

存儲 MDB 資料表和 RDB 資料表之間的同步關係以及 UserCommand。UserCommand 表示欄位之間的同步關係。具體為

DMSYNC_SYNC_OBJ

欄位	類型	注釋
OBJID	SERIAL(1) PRIMARY KEY	Objection 的 ID
PUBID	INTEGER	此 Objection 關聯的 Publication
MDB_TBL	VARCHAR(128)	表示 Objection 對應的 MDB table
RDB_TBL	VARCHAR(128)	表示 Objection 對應的 RDB table
OBJOPTION	VARCHAR(32)	兩個 table 資料衝突時採用的處理策略
UP_INS_CMD	LONG VARCHAR	上傳 Insert 資料更改執行的 UserCommand
UP_UPD_CMD	LONG VARCHAR	上傳 Update 資料更改執行的 UserCommand
UP_DEL_CMD	LONG VARCHAR	上傳 Delete 資料更改執行的 UserCommand
UP_OVR_CMD	LONG VARCHAR	ModifyTime 方式 資料衝突策略執行的 UserCommand (參見 DMSync 衝突)
DN_INS_CMD	LONG VARCHAR	下載 Insert 資料更改執行的 UserCommand
DN_UPD_CMD	LONG VARCHAR	下載 Update 資料更改執行的 UserCommand
DN_DEL_CMD	LONG VARCHAR	下載 Delete 資料更改執行的 UserCommand
DN_OVR_CMD	LONG VARCHAR	資料衝突時執行的 UserCommand (依據 OBJOPTION)

UP_UPD_QRY	LONG VARCHAR	timestamp 同步方式上傳 Insert&Update 更改時執行的 UserCommand
UP_DEL_QRY	LONG VARCHAR	timestamp 同步方式上傳 Delete 更改時執行的 UserCommand
DN_UPD_QRY	LONG VARCHAR	timestamp 同步方式下載 Insert&Update 更改時執行的 UserCommand
DN_DEL_QRY	LONG VARCHAR	timestamp 同步方式下載 Delete 更改時執行的 UserCommand
MDB_TBL_PK	VARCHAR(128)	MDB Table 主鍵的 column 名稱
RDB_TBL_PK	VARCHAR(128)	RDB Table 的主鍵 column 名稱

Objoption:00000

1. 衝突偵測 0: server, 1: modify

server: 衝突發生時，以 server data 為主 (先同步者優先)

modify: 衝突發生時，採用 RDB record modify time 時間戳優先

SQL 參考腳本

```
CREATE TABLE DMSYNC_SYNC_OBJ (
    OBJID SERIAL(1) primary key,
    PUBID INTEGER not null ,
    MDB_TBL VARCHAR(128) not null ,
    RDB_TBL VARCHAR(128) not null ,
    OBJOPTION VARCHAR(32) not null ,
    UP_INS_CMD LONG VARCHAR default null ,
    UP_UPD_CMD LONG VARCHAR default null ,
    UP_DEL_CMD LONG VARCHAR default null ,
    UP_OVR_CMD LONG VARCHAR default null ,
    DN_INS_CMD LONG VARCHAR default null ,
    DN_UPD_CMD LONG VARCHAR default null ,
    DN_DEL_CMD LONG VARCHAR default null ,
    DN_OVR_CMD LONG VARCHAR default null ,
    UP_UPD_QRY LONG VARCHAR default null ,
    UP_DEL_QRY LONG VARCHAR default null ,
    DN_UPD_QRY LONG VARCHAR default null ,
    DN_DEL_QRY LONG VARCHAR default null ,
    MDB_TBL_PK VARCHAR (128) not null ,
    RDB_TBL_PK VARCHAR (128) not null
);
```

DMSYNC_SYNC_RDB

一個 RDB 端在同步之前需要先在 DMSync Server 上註冊，這些資訊主要用於 RDB 分配相應的 ID 值並記錄 RDB 的標示和所在設備。

DMSYNC_SYNC_RDB

欄位	類型	注釋
RID	INTEGER primary key	為 RDB 所分配的唯一 ID
MACHINE_ID	VARCHAR(128)	為遠端設備分配的 ID

SQL 參考腳本

```
CREATE TABLE DMSYNC_SYNC_RDB (
    RID INTEGER NOT NULL PRIMARY KEY,
    MACHINE_ID VARCHAR(128) NOT NULL
);
```

DMSYNC_SYNC_SUB

DMSYNC_SYNC_SUB 存儲了 Subscription。

DMSYNC_SYNC_SUB

欄位	類型	注釋
RID	INTEGER	Subscription 所對應的 Remoteld
SUBID	INTEGER	為 Subscription 分配的唯一 ID
USERID	INTEGER	Subscription 所使用的 DMS_USER 用戶
PUBID	INTEGER	Subscription 所對應的 publication id
OFFSET	INTEGER	最後同步成功的位移
LAST_UP_TM	TIMESTAMP	最後成功上傳的時間
LAST_DN_TM	TIMESTAMP	最後成功下載的時間

Primary key(rid,subid) 複合主鍵

複合主鍵是因為會有多個 RDB 與一個 MDB 進行同步。

DMSYNC_SYNC_USER

這裏的 USER 是 DMSync 的 SyncUser，和 DB User 是有區別的，是使用 Connect () 連接 DMSync Server 時的 User。DMSYNC_SYNC_USER 存儲了所有註冊過的 SyncUser。

欄位	類型	注釋
USERID	INTEGER PRIMARY KEY	DMSync 用戶的 ID
USERNAME	VARCHAR(32)	DMSync 用戶的名稱
PASSWORD	CHAR(32)	DMSync 用戶的密碼
SSH_PUB	LONG VARBINARY	SSH 安全模式的 PUB KEY

Password 目前採用明文存儲

SQL 參考腳本

```
CREATE TABLE SYSADM.DMSYNC_SYNC_USER (
    USERID INTEGER NOT NULL PRIMARY KEY,
    USERNAME VARCHAR (32) NOT NULL,
    PASSWORD CHAR (32) NOT NULL,
    SSH_PUB LONG VARBINARY DEFAULT NULL
)
```

2.3 Jetty 構建

前面已經提到過 Jetty，本節我們將介紹如何配置 Jetty 使其支援 DMSync。當然需要先從官方網站下載 Jetty 和 axis2，在此我們推薦用戶使用已經在 DMSync 中測試過的版本。在確定 DMSync Server 主機後，請按以下步驟進行。

- 1) 解壓 Jetty 相關的 zip 檔到 DMSync Server 上的目錄。
 - 2) 進入 jetty\webapps 目錄,將提供的 axis2 包解壓至此目錄下,即 jetty\webapps\axis2
 - 3) 進入 jetty\webapps\axis2\WEB-INF\services 目錄,將 DMSync CD 提供的 dmsync-x.x.aar(目前版本為 dmsync-1.0.aar)放置此目錄中,即
jetty\webapps\axis2\WEB-INF\services\dmsync-1.0.aar
 - 4) 進入 jetty\webapps\axis2\WEB-INF\conf 目錄,將 DMSync CD 提供的 dmsync.xml 和 axis2.xml 放入此目錄中,dmsync.xml 包含了 DMSync DB 的位址資訊,用戶需要對此資訊進行配置,請參考 DMSync Server 部署。
 - 5) 進入 jetty\webapps\axis2\WEB-INF\lib 目錄,將 DMSync 提供的 dmjdbc30.jar, ehcache-1.4.0.jar 這兩個 jar 包放入此文件夾中。其中的 ehcache-1.4.0.jar 是 Jetty 用來配置 DMSync Server 的,dmjdbc30.jar 則是 DBMaker 的 JDBC 驅動。需要再次強調的是,將某一 RDBMS 的 JDBC Driver 放入此 lib 檔夾,並不意味著它就可以被 DMSync 支援並作為 MDB。
- 至此,DMSync Server 的構建就已經完成並可以正常啓動 Jetty 了。

2.4 Jetty DMSync.XML 配置

上文提到的 dmsync.xml 記錄了 Jetty 要使用的且對應於 DMSync DB 的 IP、埠號及 Clicode。在 dmsync.xml 中主要包含兩部分內容:cache 配置和 DMSync DB 配置。Cache 部分將在 DMSync 性能調優章節中講述,本節則講述 DMSync DB 的配置。

dmsync.xml 中的 DMSync DB 結構如下所示:

```
<DMSyncDB>
  <c3p0>
    <jdbcUrl>jdbc:dbmaker://127.0.0.1:3307/DMSYNC;clicode=UTF-8</jdbcUrl>
    <password></password>
    <user>SYSADM</user>
  </c3p0>
</DMSyncDB>
```

DMSync DB 主要由 jdbcUrl, clicode, username 和 password 幾部分組成。CLILCODE 的設定要與 LCODE 保持一致,除非 DB 的 LCODE 值為 10,即 UTF-8 類型,此時可以將 Clicode 設置為多種類型,如 GBK, BIG5 等。

具體說明如下:

- 1) IP, 需要以實際運行 DMSync DB 的 IP 位址來替換<jdbcUrl>的鍵值中的 IP 值。
這裏的 127.0.0.1 標示了 Jetty 和 DMSync DB 運行在同一主機。
- 2) 埠, 需要以實際運行的 DMSync DB 埠號來替換<jdbcUrl>埠號。
- 3) 資料庫名稱, 以實際的 DMSync DB 名稱來替換。
- 4) 用戶名和密碼, 填寫用於訪問 DMSync DB 的用戶名和密碼。

JDBC URL

dmsync.xml 中的 jdbcUrl 表示 DMSync DB 的位置和 clicode。可以看出此處的 jdbcUrl 實際上是 DBMaker JDBC 的標準 URL 寫法。另外, MDB 的 jdbcUrl 也與 DMSync DB 的 jdbcUrl 寫法類似。現在,我們可以很容易地理解 DMSync 操作 DB 也是通過 JDBC 來進行的,相應的 DB 必須有 JDBC 的支持。

通過 DBMaker 的 jdbcUrl 就可以知道 DMSync 運行所在的主機、埠和資料庫名稱。前

面章節我們已經建議用戶將 DMSync DB 命名為“DMSync”，其實 DMSync DB 只要包含 DMSync 要求的所有系統表就可以了，對於 DB 名稱並沒有強制要求，只需要在 dmsync.xml 中正確配置 DMSync DB 的 jdbcUrl 即可，如：

```
jdbc:dbmaker://10.0.0.1:3309/DMSYNC2;clilcode=utf-8
```

2.5 Jetty Server 啓動

完成以上配置後，Jetty 就可以正常啓動了。首先進入 cmd line 模式並切換至 Jetty 目錄，然後在命令行提示符後輸入以下命令，啓動 Jetty。

```
Jetty7>java -jar start.jar
```

3 DMSync GLOG 環境

DMSync 有兩種同步方式：GLOG 和 Timestamp。本手冊是以 DMSync 1.01 版本為講述物件，因為該版本對 Timestamp 方式暫不支援，所以目前只講述 GLOG 方式。

3.1 GLOG 概述

本手冊以一個聯繫人資訊同步的例子作為 TELEBOOK Sample。企業通訊錄中，員工需要經常更新聯繫人資訊。本例中，要求在 MDB 中存儲的聯繫人資訊如下表 3.1 所示。

MDB Telebook	
欄位	注釋
_ID	標識電話簿的唯一 ID
NAME	TELEBOOK 中文姓名
PHONE_NUMBER	電話號碼
AGE	年齡
COMPANY	公司名稱
DEPARTMENT	部門號
JOINDATE	入職日期
EMAIL	郵件地址

表 3.1

MDB 中存儲了聯繫人的詳細資訊，公司員工往往因為業務聯繫等原因，需要將 MDB 中的資訊下載到自己的手持設備中。此時員工的手持設備就作為遠端設備，設備上安裝的資料庫就是 RDB，可是在現實應用中往往不需要下載聯繫人的全部資訊，只需下載一部分必要的資訊，比如姓名和電話號碼等，如表 3.2。

RDB Telebook	
欄位	注釋
_ID	標識電話簿的唯一 ID
NAME	TELEBOOK 中文姓名
PHONE_NUMBER	電話號碼

表 3.2

GLOG 方式要求 MDB 和 RDB 兩端都要記錄表資料更改的前、後映射值，並記錄更改序列，如對 MDB 的 Telebook 表的任何資料操作都要記錄前後映射值。另外，雖然一般的 RDBMS 都具有日誌功能，但是要通過日誌功能來記錄值的更改顯然不方便，所以我們需要創建一個 LOG 表，此表會針對 Telebook 的每一列分別定義兩列，比如針對 Telebook 的 NAME 列在 LOG 表中創建 NAME(新值)和 ONAME (舊值)來與之對應，分別用來記錄更改前後的值，如下：

- TELEBOOK 執行 INSERT，NAME 賦值為 Alan，則 LOG 表增加一筆記錄，NAME 和 ONAME，值分別為 Alan 和 NULL；
- TELEBOOK 執行 UPDATE，更新 Alan 值 Bob，則 LOG 表增加另一筆記錄，NAME

和 ONAME，值分別為 Bob 和 Alan；

- TELEBOOK 執行 DELETE，刪除了 BOB 的資訊，則 LOG 表要增加新的一筆記錄，NAME 和 ONAME，值分別為 NULL 和 Bob。

通過使用 LOG 表就可以完整地記錄 TELEBOOK 表值的更改。LOG 表的命名規則為：**MDB 名_SYNC_LOG_數據表名**。根據這個規則，MDB TELEBOOK 的 LOG 表名稱就為 MDB_SYNC_LOG_TELEBOOK。另外需要為 TELEBOOK 創建 INSERT, UPDATE 和 DELETE 觸發器用來將記錄更改至 MDB_SYNC_LOG_TELEBOOK，觸發器的命名規則同樣也要嚴格遵循要求。

當 MDB 擁有多個資料表時，需要一個 GLOG 表來記錄 TELEBOOK 資料表和其他資料表的更改序列。GLOG 主要記錄變更的表名稱和 LOG ID，這樣 DMSync 通過 GLOG 表就可以知道資料庫中哪些表發生了變更，然後通過 LOG ID 對應到 LOG 表中獲得更改的前後值。本例 GLOG 表的完整表名稱為 MDB_SYNC_GLOG，顯然 MDB 只能擁有一個 GLOG 表。

RDB 採用同樣策略，區別在於命名規則。比如針對 RDB 的 TELEBOOK，LOG 表和 GLOG 表的表名稱分別為 RDB_SYNC_LOG_TELEBOOK 和 RDB_SYNC_GLOG。

3.2 DMSync MDB GLOG 環境

MDB(MasterDB)主資料庫也稱為中央存儲資料庫，用來維護那些來自於多個 RDB 的同步資料。MDB 必須支持 JDBC，目前僅限於 DBMaker。用戶需要先創建 MDB，再建立相應的資料同步表。DMSync 同步時資料更改是基於 column 的，所以必須考慮 column 類型之間的相容性，如 RDB (c1 varchar(5)) ->MDB (c2 char(10))。另外還需要根據 MDB 的特性來考慮資料的存儲量以及性能方面的問題，用戶可根據 GLOG 在 MDB 中創建下表中的內容。

名稱	注釋
MDB_SYNC_GLOG	記錄所有資料表的更改序列
TELEBOOK	本例中要創建的資料表
MDB_SYNC_LOG_TELEBOOK	記錄 TELEBOOK 表的更改
Trigger ON Telebook	將 TELEBOOK 執行更改的新舊值存入 LOG 表。
Trigger on MDB_SYNC_LOG_TELEBOOK	將 TELEBOOK 的更改加入 GLOG 序列

表 3.3

MDB 創建及初始化

MDB 的位置和名稱都存儲在 Publication 中，且以 JDBC URL 的形式存儲。對 MDB 資料庫名稱並不要求，http://auto.msn.com.cn/auto_guide/20110615/1253503.shtml 這裏創建為“MDB”。

```
DmSQL> Create db MDB;
```

有關配置檔 dmconfig.ini 的設定請參考 DBMaker 資料庫管理員手冊，在配置 LCODE 欄位時，應與 DMSync DB 的 LCODE 保持一致。

創建 MDB_SYNC_GLOG 系統表

GLOG 主要用來記錄 MDB 中資料更改的表名稱。

MDB_SYNC_GLOG

欄位	類型	注釋
GID	Serial Primary Key	GLOG 序列的 ID。

LOGTBL	Varchar(128)	資料變動的資料表名。
LOGTBLID	INTEGER	資料變動 LOG 表中的資料 ID。
LOGTM	TIMESTAMP	變動發生時間值。
RID	CHAR(32)	資料表對應同步表所在的 RDB。

SQL 參考腳本

```
CREATE TABLE SYSADM.MDB_SYNC_GLOG (
  GID SERIAL(1),
  LOGTBL VARCHAR(128) DEFAULT NULL ,
  LOGTBLID INTEGER DEFAULT NULL ,
  LOGTM TIMESTAMP DEFAULT NULL ,
  RID CHAR(32) DEFAULT NULL
);
```

創建 TELEBOOK 資料表

根據 TELEBOOK 表的實際需要確定 TELEBOOK 的 schema。

TELEBOOK

欄位	類型
_ID	SERIAL
NAME	CHAR(20)
PHONE_NUMBER	CHAR(50)
AGE	INT
COMPANY	CHAR(50)
DEPARTMENT	INT
JOINDATE	DATE
EMAIL	CHAR(100)

SQL 參考腳本

```
CREATE TABLE TELEBOOK (
  _ID SERIAL PRIMARY KEY,
  NAME CHAR (20),
  PHONE_NUMBER CHAR (50),
  AGE INT,
  COMPANY CHAR (50),
  DEPARTMENT INT,
  JOINDATE DATE,
  EMAIL CHAR (100)
);
```

創建 MDB_SYNC_LOG_TELEBOOK 表

如果 RDB 只需要聯繫人的部分資訊，如記錄部分欄位的新舊值時，我們建議用戶為所有欄位都記錄新舊值以備新的需求。在創建 LOG 表時，用戶可增加一個欄位來記錄資料變更是由哪個操作執行的，INSERT, UPDATE, DELETE 可分別由 0, 1, 2 這三個數字來表示。LOG 表的 Schema 如下所示：

MDB_SYNC_LOG_TELEBOOK

欄位	類型	注釋
LOGID	SERIAL PRIMARY KEY	記錄 LOG ID
CMD	INT	記錄 TELEBOOK 的操作，取值 0,1,2
_ID	INT	新值
NAME	CHAR(20)	
PHONE_NUMBER	CHAR(50)	
AGE	CHAR(20)	
COMPANY	CHAR(10)	
DEPARTMENT	CHAR(50)	
JOINDATE	CHAR(50)	
EMAIL	CHAR(20)	
O_ID	INT	舊值
ONAME	CHAR(20)	
OEXTENTION	CHAR(50)	
OENGLISH_NAME	CHAR(20)	
OUSER_ID	CHAR(10)	
OCOMPANY	CHAR(50)	
ODEPARTMENT	CHAR(50)	
OOFFICE	CHAR(20)	
OEMAIL	CHAR(50)	

SQL 參考腳本

```
CREATE TABLE MDB_SYNC_LOG_TELEBOOK(
    LOGID SERIAL PRIMARY KEY,CMD INT,
    _ID SERIAL PRIMARY KEY ,
    NAME CHAR(20),
    PHONE_NUMBER CHAR(50),
    AGE INT,
    COMPANY CHAR(50),
    DEPARTMENT INT,
    JOINDATE DATE,
    EMAIL CHAR(100),
    O_ID SERIAL PRIMARY KEY ,
    ONAME CHAR(20),
    OPHONE_NUMBER CHAR(50),
    OAGE INT,
    OCOMPANY CHAR(50),
    ODEPARTMENT INT,
    OJOINDATE DATE,
    OEMAIL CHAR(100)
);
```

創建對應的 MDB Trigger

為了記錄 LOG 變更和 GLOG 序列，需要對 TELEBOOK 的 INSERT、UPDATE 和

DELETE 操作分別創建觸發器，這三個觸發器的名稱爲：**INS_TELEBOOK, UPD_TELEBOOK, DEL_TELEBOOK**。

可以看出，這些觸發器的名稱都是由相應操作的英文簡寫和表名稱結合而成的，並且要注意區分大小寫，此規則適用於所有資料同步表。

接下來介紹這三個觸發器的創建過程。首先，在觸發器語法中用到了 NEW 和 OLD 這兩個關鍵字，它區別於 DMSync 的{\$NEW}和{\$OLD}，此處用法屬於 SQL 標準。關於觸發器相關語法的說明，請參考 DBMaker 資料庫管理員手冊。

INS_TRIGGER ON MDB TELEBOOK

觸發器 INS_TELEBOOK 用於在表 TELEBOOK 中執行 INSERT 操作時，將新值插入到 LOG 表中，INSERT 用 0 來表示，詳細描述如下。

TELEBOOK VALUES		MDB_SYNC_LOG_TELEBOOK
NEW._ID	→→	_ID
NEW.NAME	→→	NAME
NEW.PHONE_NUMBER	→→	PHONE_NUMBER
NEW.AGE	→→	AGE
NEW.COMPANY	→→	COMPANY
NEW.DEPARTMENT	→→	DEPARTMENT
NEW.JOINDATE	→→	JOINDATE
NEW.EMAIL	→→	EMAIL

SQL 參考腳本

```
CREATE TRIGGER INS_TELEBOOK AFTER INSERT ON TELEBOOK FOR EACH ROW(
    INSERT INTO MDB_SYNC_LOG_TELEBOOK (
        LOGID,
        CMD,
        _ID,
        NAME,
        PHONE_NUMBER,
        AGE,
        COMPANY,
        DEPARTMENT,
        JOINDATE,
        EMAIL
    ) VALUES (
        NULL,
        0,
        NEW._ID,
        NEW.NAME,
        NEW.PHONE_NUMBER,
        NEW.AGE,
        NEW.COMPANY,
        NEW.DEPARTMENT,
```

```

NEW.JOINDATE,
NEW.EMAIL
)
);

```

UPD_TRIGGER ON MDB TELEBOOK

觸發器 UPD_TRIGGER 用於在表 TELEBOOK 中執行 UPDATE 操作時，將更新前的 old 值和更新後的新值記錄到 LOG 表中，UPDATE 用 1 表示。

TELEBOOK VALUES		MDB_LOG_TELEBOOK
NEW._ID	→→	_ID
NEW.NAME	→→	NAME
NEW.PHONE_NUMBER	→→	PHONE_NUMBER
NEW.AGE	→→	AGE
NEW.COMPANY	→→	COMPANY
NEW.DEPARTMENT	→→	DEPARTMENT
NEW.JOINDATE	→→	JOINDATE
NEW.EMAIL	→→	EMAIL
OLD._ID	→→	O_ID
OLD.NAME	→→	ONAME
OLD.PHONE_NUMBER	→→	OPHONE_NUMBER
OLD.AGE	→→	OAGE
OLD.COMPANY	→→	OCOMPANY
OLD.DEPARTMENT	→→	ODEPARTMENT
OLD.JOINDATE	→→	OJOINDATE
OLD.EMAIL	→→	OEMAIL

SQL 參考腳本

```

CREATE TRIGGER UPD_TELEBOOK AFTER UPDATE ON TELEBOOK FOR EACH ROW(
    INSERT INTO MDB_SYNC_LOG_TELEBOOK(
        LOGID,
        CMD,
        _ID,
        NAME,
        PHONE_NUMBER,
        AGE,
        COMPANY,
        DEPARTMENT,
        JOINDATE,
        EMAIL,
        O_ID,
        ONAME,
        OPHONE_NUMBER,
        OAGE,

```

```

        OCOMPANY,
        ODEPARTMENT,
        OJOINDATE,
        OEMAIL
    ) VALUES (
        NULL,
        1,
        NEW._ID,
        NEW.NAME,
        NEW.PHONE_NUMBER,
        NEW.AGE,
        NEW.COMPANY,
        NEW.DEPARTMENT,
        NEW.JOINDATE,
        NEW.EMAIL,
        OLD._ID,
        OLD.NAME,
        OLD.PHONE_NUMBER,
        OLD.AGE,
        OLD.COMPANY,
        OLD.DEPARTMENT,
        OLD.JOINDATE,
        OLD.EMAIL
    )
);

```

DEL_TRIGGER ON MDB TELEBOOK

觸發器 DEL_TRIGGER 用於在表 TELEBOOK 中執行 DELETE 操作時，記錄被刪除的值，DELETE 用 2 表示。

TELEBOOK VALUES		MDB_LOG_TELEBOOK
OLD._ID	→→	O_ID
OLD.NAME	→→	ONAME
OLD.PHONE_NUMBER	→→	OPHONE_NUMBER
OLD.AGE	→→	OAGE
OLD.COMPANY	→→	OCOMPANY
OLD.DEPARTMENT	→→	ODEPARTMENT
OLD.JOINDATE	→→	OJOINDATE
OLD.EMAIL	→→	OEMAIL

SQL 參考腳本

```

CREATE TRIGGER DEL_TELEBOOK BEFORE DELETE ON TELEBOOK FOR EACH ROW(
    INSERT INTO MDB_SYNC_LOG_TELEBOOK(

```



```

        LOGID,
        CMD,
        O_ID,
        ONAME,
        OPHONE_NUMBER,
        OAGE,
        OCOMPANY,
        ODEPARTMENT,
        OJOINDATE,
        OEMAIL
    ) VALUES (
        NULL,
        2,
        OLD._ID,
        OLD.NAME,
        OLD.PHONE_NUMBER,
        OLD.AGE,
        OLD.COMPANY,
        OLD.DEPARTMENT,
        OLD.JOINDATE,
        OLD.EMAIL
    )
);

```

INS TRIGGER ON MDB_SYNC_LOG_TELEBOOK

剛才在 TELEBOOK 表上創建的 3 個觸發器，能夠保證所有動作引起的資料更改值都記錄在 MDB_SYNC_LOG_TELEBOOK 中，此時仍然將 TELEBOOK 的 LOG 更改序列保存到 GLOG 表裏，供同步時檢索。

TELEBOOK 執行 IUD 都會產生 LOG，而這些操作是由用戶執行的；而 LOG 表 MDB_SYNC_LOG_TELEBOOK 是 DMSync 要求的附加表，操作是 DMSync 執行的。在 LOG 表上創建一個 INSERT 類型的 Trigger 就可以為每條 LOG 資訊產生 GLOG 序列。

對 LOG 上的命名也有要求，INS_SYNC_LOG_TABLENAME。本例中創建的 INSERT 類型的觸發器是 INS_SYNC_LOG_TELEBOOK。

MDB_SYNC_LOG_TELEBOOK VALUES		MDB_SYNC_GLOG
NULL(auto increment id)	→→	GID
TABLENAME(數據表名，如 TELEBOOK)	→→	LOGTBL
LOGTABLEID(LOGID)	→→	LOGTBLID
Now()(記錄變更時間)	→→	LOGTM
GETRID()(系統 RDB 獲取函數)	→→	RID

Getrid()是 DBMaker 的自定義函數，需要 DBMaker 5.2 以上版本才能支持。

SQL 參考腳本

```

CREATE TRIGGER INS_SYNC_LOG_TELEBOOK
AFTER INSERT ON MDB_SYNC_LOG_TELEBOOK FOR EACH ROW (

```

```

INSERT INTO MDB_SYNC_GLOG VALUES (
    NULL,
    'TELEBOOK',
    NEW.LOGID,
    NOW(),
    GETRID()
);

```

至此 MDB 的環境就配置完成，但 DMSync 為什麼要對觸發器的命名進行限制呢？因為 DMSync 是通過 JDBC 來操作資料庫的，在進行同步時，要刪除那些名稱符合約定的觸發器。如果不處理觸發器，當 DMSync 將 RDB TELEBOOK 同步來的資料通過 JDBC 插入到 MDB TELEBOOK 時，觸發器就會生成對應的 LOG 和 GLOG 記錄，DMSync 還會將同樣的資料傳輸到 RDB 端……這樣顯然是不行的。因為傳輸來的資料並不能通過觸發器產生 GLOG 序列，GLOG 序列會致使 DMSync 將傳輸來的資料再傳回去。因此，DMSync 在通過 JDBC 插入資料時，如果資料來源為同步過來的，那麼 DMSync 會將所有觸發器刪除，完成後再回復創建的觸發器，該過程無需用戶的干預。

3.3 DMSync RDB GLOG 環境

配置好 MDB GLOG 環境後，本節我們就開始講述 RDB 環境的構建了。這裏的 RDB 是指運行在 Android 上的 SQLite3，有關 Android 的詳細情況請參考 <http://www.android.com>。

對於 RDB，目前僅以 GLOG 方式的例子來演示，其環境的構建方法與 MDB 類似。不過由於 RDB 也是作為 DMSync Server 的 Client 端，所以除了為 GLOG 方式創建相關的表外，還需要創建 DMSync Client 的系統表。

本例對 RDB 資料表 TELEBOOK 的設計是，它只與 MDB 資料表 TELEBOOK 中的部分欄位對應做同步。

RDB 創建及初始化

在 SQLite3 中創建資料庫時，只需要在 shell 中執行 Sqlite3 DBName。如下例在 adb Shell 中創建一個 DBName 為“rdb”的資料庫。

```
~# sqlite3 rdb
```

因為 RDB 是獨立運行在遠端設備上的，所以在 RDB 初始化時不僅要創建 GLOG 表，還要創建其他系統表，詳細資訊如下：

名稱	注釋
RDB_SYNC_RSUB	存儲 RemoteSubscription 資訊
RDB_SYNC_SYS	存儲 DMSync Client 的配置資訊，如數據包大小等
RDB_SYNC_GLOG	記錄 RDB 所有資料表的更改序列
TELEBOOK	本例中要創建的資料表
RDB_SYNC_LOG_TELEBOOK	記錄 TELEBOOK 表的更改
Trigger ON Telebook	將 TELEBOOK 執行更改的新舊值存入 LOG 表。
Trigger on RDB_SYNC_LOG_TELEBOOK	將 TELEBOOK 的更改加入 GLOG 序列

接下來需要在“RDB”資料庫中創建如下這些表。

創建 RDB_SYNC_RSUB 系統表

RDB_SYNC_RSUB 是用來記錄 RemoteSubscription 的，RSUB 中存儲了 DMSync User 的資訊，當用戶連接至 DMSync Server 時，如果沒有在 ConnectStr 中指定用戶名，那麼 DMSync 會根據 PUBName 來搜索 RSUB，並將 RSUB 中的用戶作為默認用戶。

欄位	類型	注釋
RSUBID	INTEGER PRIMARY KEY	RSUB 的 ID
DMS_USER	VARCHAR(128)	默認 DMSync 用戶名
DMS_URL	VARCHAR(128)	DMSync Server 地址
DMS_PASSWORD	VARCHAR(128)	默認 DMSync 用戶密碼
DMS_TYPE	VARCHAR(128)	DMSync MDB 類型 (目前為 DBMaker)
PUBNAME	VARCHAR(128)	RSUB 對應的 PUB
OFFSET	INT	最後成功下載的位移
LAST_UP_TM	TIMESTAMP	最後成功上傳的時間
LAST_DN_TM	TIMESTAMP	最後成功下載的時間

SQL 參考腳本

```
CREATE TABLE RDB_SYNC_RSUB(
    RSUBID INTEGER PRIMARY KEY,
    DMS_USER VARCHAR(128) NOT NULL,
    DMS_URL VARCHAR(128) NOT NULL,
    DMS_PASSWORD VARCHAR(128) NOT NULL,
    DMS_TYPE VARCHAR(128) NOT NULL,
    PUBNAME VARCHAR(128) NOT NULL,
    OFFSET INT NOT NULL,
    LAST_UP_TM TIMESTAMP,
    LAST_DN_TM TIMESTAMP
);
```

創建 RDB_SYNC_SYS 系統表

RDB_SYNC_SYS 用來存放 RDB 端作為 DMSyncClient 的設置資訊，如 uploadpacketize 等，這些設置資訊以記錄的形式寫進 RDB_SYNC_SYS 表，key 代表設置項，value 代表設置的值。

RDB_SYNC_SYS

欄位	類型	注釋
KEY	VARCHAR(32) PRIMARY KEY	設置的鍵
VALUE	VARCHAR(128)	設置的鍵值

SQL 參考腳本

```
CREATE TABLE RDB_SYNC_SYS(
    KEY VARCHAR(32) PRIMARY KEY,
    VALUE VARCHAR(128) NOT NULL
);
```

Key 的詳細設置將在後面章節進行講述。

創建 RDB_SYNC_GLOG 系統表

RDB_SYNC_GLOG 用來記錄 RDB GLOG 資料表的變更序列。

RDB_SYNC_GLOG

欄位	類型	注釋
GID	Serial Primary Key	RDB GLOG 序列 ID。
LOGTBL	Varchar(128)	RDB 發生變動的資料表名
LOGTBLID	INTEGER	RDB LOG 表的 LOG ID
LOGTM	TIMESTAMP	RDB LOG 產生的時間。
DONE	CHAR(1)	RDB 保留

SQL 參考腳本

```
CREATE TABLE RDB_SYNC_GLOG(
  GID INTEGER PRIMARY KEY,
  LOGTBL VARCHAR(128),
  LOGTBLID INT,
  LOGTM TIMESTAMP,
  DONE CHAR(1)
);
```

創建 RDB_TELEBOOK 資料表

與 MDB 的同步表 TELEBOOK 不同的是，RDB 創建的 TELEBOOK 只包含少量欄位。由於 SQLite 僅支持 5 種資料類型，所以 RDB TELEBOOK 和 MDB TELEBOOK 對應的欄位類型幾乎不可能一樣。這也說明了 DMSync 的同步是基於 column，而對表的 column 總數和類型是沒有要求的，不過在設置時仍然要考慮不同類型之間的相容性。此處就是用 RDB TELEBOOK 的 PHONE_NO 欄位和 MDB TELEBOOK 的 PHONE_NUMBER 欄位來進行對應的。

TELEBOOK

欄位	類型	注釋
_ID	INTEGER PRIMARY KEY	標識電話簿的唯一 ID
NAME	TEXT	TELEBOOK 中文姓名
PHONE_NO	TEXT	電話號碼

SQL 參考腳本

```
CREATE TABLE TELEBOOK(
  _ID INTEGER PRIMARY KEY AUTOINCREMENT,
  NAME TEXT,
  PHONE_NO TEXT
);
```

創建 RDB_SYNC_LOG_TELEBOOK 表

RDB TELEBOOK 表比較簡單，對應的 LOG 表和 MDB 的要求一樣，其命名形式必須為 RDB_SYNC_LOG_TABLENAME。例如 RDB TELEBOOK 的 LOG 表的名稱為：

RDB_SYNC_LOG_TELEBOOK。

欄位	類型	注釋
----	----	----

LOGID	INTEGER PRIMARY KEY	記錄 LOG ID
CMD	INT	記錄 TELEBOOK 的操作，取值 0,1,2
_ID	INT	新值
NAME	TEXT	
PHONE_NUMBER	TEXT	
O_ID	INT	舊值
ONAME	TEXT	
OEXTENTION	TEXT	

SQL 參考腳本

```
CREATE TABLE RDB_SYNC_LOG_TELEBOOK (
    LOGID INTEGER PRIMARY KEY,
    CMD INT,
    _ID INT,
    NAME TEXT,
    PHONE_NO TEXT,
    O_ID INT,
    ONAME TEXT,
    OPHONE_NO TEXT
);
```

創建 RDB 中相應的觸發器

RDB GLOG 方式也要求使用 Trigger 把資料表的更改記錄到 LOG 表中，對觸發器的命名同樣有嚴格要求。RDB TELEBOOK 需要創建的三個觸發器，名稱分別是：

INS_TELEBOOK, UPD_TELEBOOK, DEL_TELEBOOK。

在 RDB_SYNC_LOG_TELEBOOK 上創建的觸發器是：

INS_SYNC_LOG_TELEBOOK。

SQLite3 創建觸發器的語法和 DBMaker 類似，詳細資訊請參考 SQLite 官方資料。

INS_TRIGGER ON RDB TELEBOOK :

在 RDB 中對 TELEBOOK 進行 INSERT 操作需要記錄插入後的 new 值。INSERT 操作用 0 來表示，名稱爲 INS_TABLENAME。

TELEBOOK VALUES		RDB_SYNC_LOG_TELEBOOK
NEW._ID	→→→	_ID
NEW.NAME	→→	NAME
NEW.PHONE_NO	→→	PHONE_NO

SQL 參考腳本

```
CREATE TRIGGER INS_TELEBOOK AFTER
INSERT ON TELEBOOK FOR EACH ROW
BEGIN
    INSERT INTO RDB_SYNC_LOG_TELEBOOK
        (LOGID,CMD,_ID,NAME,PHONE_NO)
    VALUES
```

```
(NULL,0,NEW._ID,NEW.NAME,NEW.PHONE_NO);
END;
```

UPD_TRIGGER ON RDB TELEBOOK

在 RDB 中對 TELEBOOK 進行 UPDATE 操作用 1 來表示，名稱爲 UPD_TABLENAME。

TELEBOOK		RDB_LOG_TELEBOOK
NEW._ID	→→	_ID
NEW.NAME	→→	NAME
NEW.PHONE_NO	→→	PHONE_NO
OLD._ID	→→	O_ID
OLD.NAME	→→	ONAME
OLD.PHONE_NO	→→	OPHONE_NO

可以在 TELEBOOK 表上添加 **UPD_TRIGGER** 來實現 ???

SQL 參考腳本

```
CREATE TRIGGER UPD_TELEBOOK AFTER
UPDATE ON TELEBOOK FOR EACH ROW
BEGIN
INSERT INTO RDB_SYNC_LOG_TELEBOOK
(LOGID, CMD, _ID, O_ID, NAME, ONAME, PHONE_NO, OPHONE_NO)
VALUES
(NULL, 1, NEW._ID, OLD._ID, NEW.NAME, OLD.NAME, NEW.PHONE_NO, OLD.PHONE_N
O);
END;
```

DEL_TRIGGER ON RDB TELEBOOK

在 RDB 中對 TELEBOOK 進行 DELETE 操作用 0 來表示，創建 DEL_TELEBOOK 用來記錄被刪除的值，如圖所示：

TELEBOOK		RDB_SYNC_LOG_TELEBOOK
OLD._ID	→→	O_ID
OLD.NAME	→→	ONAME
OLD.PHONE_NO	→→	OPHONE_NO

SQL 參考腳本

```
CREATE TRIGGER DEL_TELEBOOK BEFORE
DELETE ON TELEBOOK FOR EACH ROW
BEGIN
INSERT INTO
RDB_SYNC_LOG_TELEBOOK(LOGID,CMD,O_ID,ONAME,OPHONE_NO)
VALUES (NULL,2,OLD._ID,OLD.NAME,OLD.PHONE_NO);
END;
```

INS TRIGGER ON RDB_SYNC_LOG_TELEBOOK

在 RDB 上需要使用相同的方法來記錄資料變更的序列，不同的是 rid 列無需處理。

RDB_LOG_TELEBOOK		RDB_SYNC_GLOG
GID		NULL(auto increment id)
LOGTBL	→→	TABLENAME(產生變更的表名，如 TELEBOOK)
LOGTBLID	→→	LOGTABLEID(log 表中數據變更的 ID)
LOGTM		Now()(記錄變更時間)
RID		NULL(無需處理)

SQL 參考腳本

```
CREATE TRIGGER INS_SYNC_LOG_TELEBOOK AFTER INSERT ON
RDB_SYNC_LOG_TELEBOOK FOR EACH ROW BEGIN
  INSERT INTO RDB_SYNC_GLOG VALUES (
    NULL,
    'TELEBOOK',
    NEW.LOGID,
    DATETIME('NOW'),
    NULL);
END;
```

4 部署 DMSync 同步

完成 DMSync Server、MDB 和 RDB 的環境建制後，接下來就可以部署 DMSync 配置並開始使用同步了。

前面章節已經對 publication, subscription 和 objection 的概念和作用進行了介紹，下面我們會講述如何增加 publication, subscription 和 objection 的資訊。

在後序的 DMSync 版本中，我們會提供一系列 GUI 工具來幫助用戶完成相關設置，實際上這些工具的作用只是在已經創建好的系統表中增加一些有對應關係的資料記錄，另外對於 PUB, SUB 和 OBJ 來說，都會有對應的 ID 作為相應的依據。

PUB 可以確定 MDB 的位置，PUB 對應的 SUB 可以確定 RDB 的位置和使用的 USER，PUB 對應的 OBJ 可以確定 MDB 的資料表和 RDB 的資料表，OBJ 中的 UserCommand 可以確定兩表中欄位的對應關係。

4.1 增加 DMSYNC USER

DMSync User 存儲在 DMSYNC_SYNC_USER 系統表中，每增加一條記錄，相當於增加了一個 DMSync 用戶，此處為 TELEBOOK 的例子增加了兩個用戶。對於 DMSync User 的密碼而言，目前版本僅支援以明文形式存儲。

欄位	值 1	值 2
USERID	1	2
USERNAME	user1	user2
PASSWORD	123456	abcdef
SSH_PUB	空	

注：SSH_PUB：目前暫不支持

用戶名和密碼對大小寫敏感

通過 dmSQL，在表 DMSYNC_SYNC_USER 中插入兩條記錄。

SQL 參考腳本

```
dmSQL> insert into SYSADM.DMSYNC_SYNC_USER values (1, 'user1', '123456', NULL);
dmSQL> insert into SYSADM.DMSYNC_SYNC_USER values (2, 'user2', 'abcdef', NULL);
```

4.2 註冊遠端資料庫 RDB

在 DMSync RDB GLOG 環境中，創建名為“rdb”的資料庫(注意大小寫)，在 DMSYNC_SYNC_RDB 中插入記錄後，意味著 rdb 已經註冊可以參與該 DMSync Server 的同步。

在 TELEBOOK 例子中添加“rdb”資料庫。

RID	MACHINE_ID
1	Rdb

強調：rdb 是指遠端設備 sqlite3 上創建的資料庫檔。

SQL 參考腳本

```
INSERT INTO SYSADM.DMSYNC_SYNC_RDB VALUES (1, 'rdb');
```

4.3 增加 PUBLICATION

PUB 存儲在 DMYNC_SYNC_PUB 中，需要注意的是 MDB 運行主機的情況和該 PUB 的 PUBOPTION 選項設置。在 DMSync MDB GLOG 中，創建名為“MDB”的 DBMaker 資料庫，增加 PUB 時需要確認它運行所在的主機和埠號，並且用 DBMaker JDBC URL 的方式來表示。

在 TELEBOOK 的例子中，MDB 的 URL 如下：

```
JDBC:DBMAKER://127.0.0.1:3308/MDB;
```

替換具體 IP 和埠，這裏的 127.0.0.1 表示 MDB 和 DMSync DB 運行在同一主機

PUBOPTION 由五個數位組成，本例中的第一位為 0，表示通過 Transaction 的方式來傳輸資料以確保每次同步資料的完整性；第二位 0 表示同步時既要上傳 RDB 的更改，也要下載 MDB 的更改；第三位元 0 表示以 GLOG 的方式，其餘兩個保留位也使用 0 表示，那麼可以確定 PUBOPTION 串的值為 00000。

另外 Pubname 作為 ConnectString 的參數，命名要具有代表性，同時要注意區分大小寫。

本例可以確定一個 PUB 的詳細資訊。

欄位	值
PUBID	33
PUBNAME	'TELEBOOKPUB'
MDB_URL	'JDBC:DBMAKER://127.0.0.1:3308/MDB;'
MDB_USERNAME	'SYSADM'
MDB_PASSWORD	''
PUBOPTION	'00000'

Pubid 的值會在 sub 中使用，因此增加 PUB 時應先確定該 id 值，如 33。

SQL 參考腳本

```
INSERT INTO DMSYNC_SYNC_PUB (
    PUBID,
    PUBNAME,
    MDB_URL,
    MDB_USERNAME,
    MDB_PASSWORD,
    PUBOPTION)
VALUES (
    33,
    'TELEBOOKPUB',
    'JDBC:DBMAKER://127.0.0.1:3308/MDB;',
    'SYSADM',
    '',
    '00000'
);
```

4.4 增加 SUBSCRIPTION

前面幾個步驟已經增加了 DMSync 的 User 和 PUB，並註冊了一個 RDB，對應地分配了 USERID=1, PUBID=33, RID=1。

SUB 存儲在 DMSYNC_SYNC_SUB 表中，在 TELEBOOK 的例子中確定了 SUBID 為 63，則完整的 SUB 資訊如下表所示：

欄位	值
RID	1
SUBID	63
USERID	1
PUBID	33
OFFSET	0
LAST_UP_TM	'2010-11-02 11:11:22'
LAST_DN_TM	'2010-11-02 11:11:21'

從表中可以看出，當 SUB 確定後，對應的 RDB, PUB 和 USER 的對應關係就確定了。

SQL 參考腳本

```
INSERT INTO DMSYNC_SYNC_SUB (
    RID,
    SUBID,
    USERID,
    PUBID,
    OFFSET,
    LAST_UP_TM,
    LAST_DN_TM)
VALUES (
    1,
    63,
    1,
    33,
    0,
    '2010-11-02 11:11:22',
    '2010-11-02 11:11:21'
);
```

4.5 增加 USER COMMAND 和 OBJECTION

Objection 是用來確定映射關係的，每一個 OBJ 都關聯到一個 PUB，USER COMMAND 是作為 OBJ 的一部分來存儲的。

USER COMMAND

USER COMMAND 是指 DMSync 的用戶命令，由用戶自行編寫。GLOG 方式中共要編寫 6 個 USER COMMAND，即為 UPLOAD 情況下的 INSERT, UPDATE, DELETE 操作

的 USER COMMAND 以及 DOWNLOAD 情況下的 INSERT,UPDATE,DELETE。

對於資料表的每一次更改所對應的 TRIGGER 都會在 LOG 表中產生一條記錄，用來記錄操作的類型和前後映射值。同時因為 LOG 表的 TRIGGER，資料表操作也會級聯產生一條 GLOG 序列。DMSync 同步時會依據 GLOG 序列檢索 LOG 表，以獲得資料表的操作類型和前後映射值，再替換 USER COMMAND 中的運算式和變數，最終生成能在另一端資料庫執行的 SQL 語句。

DMSync 約定了一些運算式、變數和常量，USER COMMAND 中需要用到的是：

{\$NEW.ColumnName}

意味著運算式的終值是來自另一端資料庫的，\$NEW 表示更改後的值。

{\$OLD.ColumnName}

同樣表示終值是另一端資料庫對應表上指定 column 更改前的值。

DMSync 中的 USER COMMAND 和標準的 SQL 語句類似，區別在於 USER COMMAND 中的取值運算式。

所有 UPLOAD 類型的 USER COMMAND 是在 MDB 中執行的，那麼 USER COMMAND 中的 columns 部分取自 MDB 資料表，而 values 部分取自 RDB 資料表，並通過上面的兩個運算式來表示。

相應地，所有 DOWNLOAD 類型的 USER COMMAND 是在 RDB 中執行的，其 columns 部分取自 RDB 資料表，values 取自 MDB 數據表。

前面章節特意使用了 MDB TELEBOOK 的 PHONE_NUMBER 與 RDB TELEBOOK 的 PHONE_NO 兩個名稱不一樣的欄位來幫助用戶識別 TELEBOOK 的位置。RDB TELEBOOK 與 MDB TELEBOOK 的欄位之間對應映射關係如下表所示。

RDB TELEBOOK		MDB TELEBOOK
_ID	←————→	_ID
NAME	←————→	NAME
PHONE_NO	←————→	PHONE_NUMBER

接下來創建 TELEBOOK 例子中 UPLOAD 類型的 3 個 USER COMMAND。

UPLOAD INSERT USER COMMAND

UPLOAD INSERT USER COMMAND 對應於 OBJ 中的 UP_INS_CMD 欄位。當用戶在 RDB 的 TELEBOOK 中使用 INSERT 語句插入一條資料，再使用 DMSync 將其同步到 MDB 的資料表 TELEBOOK 時，DMSync 就會調用 OBJ 中的 UPLOAD_INS_CMD。

UPLOAD_INS_CMD 的 column 為 MDB TELEBOOK TABLE，values 為 RDB TELEBOOK TABLE，並且針對 INSERT，使用的插入的值，即使用表運算式{\$NEW.column}。

UP_INS_CMD:

```
INSERT INTO TELEBOOK (_ID,NAME,PHONE_NUMBER)
VALUES ({$NEW._ID},{$NEW.NAME},{$NEW.PHONE_NO})
```

從 PHONE_NO 和 PHONE_NUMBER 可以區分出此處 USER COMMAND 的 column 部分為 MDB TELEBOOK，value 部分為 RDB TELEBOOK。

UPLOAD UPDATE USER COMMAND

UPLOAD UPDATE USER COMMAND 對應於 OBJ 中的 UP_UPD_CMD 欄位，完成 UPDATE 更改的同步，比如當用戶在 RDB 的 TELEBOOK 更新剛插入的那條記錄後要執行

DMSync 同步，把已經同步到 MDB 的記錄也進行更新。

資料表必須擁有主鍵，這樣才能根據主鍵來編寫 UPDATE 相關的 USER COMMAND。為了保證同步的完整性，要比對 OLD 值，在 TELEBOOK 中，DMSync 要求當 RDB TELEBOOK LOG 的舊值與 MDB TELEBOOK 的當前值保持一致時，才能使用 RDB TELEBOOK 的當前值來更新 MDB TELEBOOK。另外同步的記錄中有可能包含空值，因此需要增加對空值的判斷處理。

UPDATE USER COMMAND 會使用到運算式{\$NEW.column}和{\$OLD.column}。

```
UP_UPD_CMD:
UPDATE TELEBOOK SET
  _ID={$NEW._ID},NAME={$NEW.NAME},PHONE_NUMBER={$NEW.PHONE_NO}
WHERE
  _ID={$OLD._ID} AND (NAME={$OLD.NAME} OR NAME IS NULL)
  AND (PHONE_NUMBER={$OLD.PHONE_NO} OR PHONE_NUMBER IS NULL) ',
_ID 是主鍵，不可能存在空值，因此無需判斷為空的情況。
```

UPLOAD DELETE USER COMMAND

UPLOAD DELETE USER COMMAND 對應於 OBJ 中的 UP_DEL_CMD 欄位，當用戶在 RDB TELEBOOK 中刪除一條記錄時，也需要把 MDB TELEBOOK 中對應的記錄刪除。在刪除記錄時，需要 RDB TELEBOOK 中的主鍵和 MDB TELEBOOK 的主鍵保持一致，此例即表示一樣的 ID 值。

```
UP_DEL_CMD:
DELETE FROM TELEBOOK WHERE _ID={$OLD._ID}
```

繼續創建 DOWNLOAD 類型的 3 個 USER COMMAND，除了 column 和 values 不一樣外，DOWNLOAD 與 UPLOAD 的寫法和用途是一樣的。這裏只給出 TELEBOOK DOWNLOAD 類型的 USER COMMAND，不作過多解釋。

DOWNLOAD INSERT USER COMMAND

Column 部分為 RDB TELEBOOK，Value 部分為 MDB TELEBOOK。

```
DN_INS_CMD:
INSERT INTO TELEBOOK (_ID,NAME,PHONE_NO)
VALUES ({$NEW._ID}, {$NEW.NAME},{$NEW.PHONE_NUMBER})
```

DOWNLOAD UPDATE USER COMMAND

```
DN_UPD_CMD:
UPDATE TELEBOOK SET
  _ID={$NEW._ID},NAME={$NEW.NAME},PHONE_NO={$NEW.PHONE_NUMBER}
WHERE
  _ID={$NEW._ID} AND (NAME={$OLD.NAME} OR NAME IS NULL)
  AND (PHONE_NO={$OLD.PHONE_NUMBER} OR PHONE_NO IS NULL)
```

DOWNLOAD DELETE USER COMMAND

```
DN_DEL_CMD:
```

```
DELETE FROM TELEBOOK WHERE _ID=({$OLD._ID})
```

通過前面所創建的 DMSync_SYNC_OBJ 表可以看出 DMSync GLOG 方式中還需要編寫衝突相關的 USER COMMAND，這部分內容將在 DMSync 衝突章節中詳細介紹，目前先假定 TELEBOOK 例子中不會發生衝突。

OBJECTION

USER COMMAND 作為 OBJ 的一部分存儲在 DMSync_SYNC_OBJ 系統表中，OBJ 中的其他內容如下表所示：

欄位	值
OBJID	33
PUBID	33
MDB_TBL	TELEBOOK
RDB_TBL	TELEBOOK
OBJOPTION	00000
USER COMMAND s	將編寫的 USER COMMAND 作為對應 column 的值 UP_OVR_CMD、DN_OVR_CMD 及 TIMESTAMP 方式預留 USER COMMAND 置空。
MDB_TBL_PK	_ID
RDB_TBL_PK	_ID

注：OBJOPTION 是決定衝突策略的，詳細資訊會在 DMSync 衝突章節中介紹。

SQL 參考腳本

```
INSERT INTO
DMSYNC_SYNC_OBJ(OBJID,PUBID,MDB_TBL,RDB_TBL,OBJOPTION,UP_INS_CMD,UP_UPD_
CMD,UP_DEL_CMD,UP_OVR_CMD,DN_INS_CMD,DN_UPD_CMD,DN_DEL_CMD,DN_OVR_CM
D,UP_UPD_QRY,UP_DEL_QRY,DN_UPD_QRY,DN_DEL_QRY,MDB_TBL_PK,RDB_TBL_PK)
VALUES(
33,
33,
'TELEBOOK',
'TELEBOOK',
'00000',
'使用 UP_INS_CMD 填充',
'使用 UP_UPD_CMD 填充',
'使用 UP_DEL_CMD 填充',
'使用 UP_OVR_CMD 填充',
'使用 DN_INS_CMD 填充',
'使用 DN_UPD_CMD 填充',
'使用 DN_DEL_CMD 填充',
'使用 DN_OVR_CMD 填充',
''''',
''''',
'_ID','_ID'
);
```

注：USER COMMAND 是作為一個長串值存入的

4.6 增加 RDB Subscription 訂閱

在 DMSync DB 中添加完 Publication 和 Subscription 以及其他相關內容後，DMSync Server 就已經準備就緒，此時可以在 RDB 端訂閱 DMSync Subscription。當一個 RDB 訂閱 Subscription 時，會生成對應的 RSUB 且存放在 RDB 端的 RDB_SYNC_RSUB 系統表中。

那麼，根據前面已經增加的資訊，要 RDB 的 RSUB 正確配置是非常簡單的。只需把 DMSync Server 端的對應資訊記錄在 RSUB 中供其使用。RDB TELEBOOK 的 RSUB 資訊如下：

欄位	值
RSUBID	63
DMS_USER	"user1"
DMS_PASSWORD	"123456"
DMS_URL	"127.0.0.1"
DMS_TYPE	"DBMAKER"
PUBNAME	"TELEBOOKPUB"
OFFSET	0
LAST_UP_TM	NULL
LAST_DN_TM	NULL

因為 ConnectStr 與 RSUB 緊密相關，RSUB 中並沒有給出 DMSync Server 的具體位址，所以建議用戶在 ConnectStr 中指出 DMSync Server 的具體位址。如果用戶沒有給出完整位址，那麼 DMSync 會根據已知參數搜索 RSUB 以補充缺少的資訊。

SQL 參考腳本

```
INSERT INTO RDB_SYNC_RSUB (
    RSUBID,
    DMS_USER,
    DMS_PASSWORD,
    DMS_URL,
    DMS_TYPE,
    PUBNAME,
    OFFSET,
    LAST_UP_TM,
    LAST_DN_TM)
VALUES(
    63,
    "user1",
    "123456",
    "127.0.0.1",
    "DBMAKER",
    "TELEBOOKPUB",
    0,
    NULL,
    NULL);
```

4.7 RDB 的其他配置

RDB_SYNC_SYNC 中存儲著 RDB 為 DMSync 系統配置的一些資訊。

關鍵字	值	
rid	1	配置已經註冊的 RDB 資訊
uploadpacketsize	4096	上傳時 Xml 包大小
servicenamespace	http://v10.operation.server.dmsync.syscom	服務命名空間

SQL 參考腳本

```
Insert into RDB_sync_sys values('uploadpacketsize','4096');  
Insert into RDB_sync_sys values('servicenamespace','http://v10.operation.server.dmsync.syscom');  
Insert into RDB_sync_sys values('rid','1');
```

5 使用 DMSync 同步

前面已經介紹過 `init()`, `connect()`, `sync()` 和 `disconnect()` 這幾個 DMSync 的 API，且這些 API 都需要在程式中調用。在學習本章之前，用戶需要具有 Android 平臺的開發經驗。

5.1 數據準備

在 TELEBOOK 例子中，向 RDB 中插入、更改、刪除資料都會生成 GLOG 序列，DMSync 可以根據這些序列將資料同步到 MDB TELEBOOK 中。另外 DMSync 要求用戶在處理資料時必須遵循一定的原則：

維護唯一的主鍵

DMSync 要求用戶創建的資料表必須擁有主鍵，並且當一條記錄插入到資料表後，不允許用戶再對其主鍵進行更改。

盡量避免衝突

設計 DMSync 資料同步表時，應避免 MDB 和 RDB 產生的非同筆資料具有相同的主鍵。

5.2 DMSync 進行同步

在進行同步之前，要保證 DMSync DB 和 MDB 都處於正常運行狀態，Jetty&&Axis2 Server 提供正常的 Web Services。DBMaker 可以通過 dmSQL 連接測試 Jetty 和 Axis2，也可以在瀏覽器中輸入 <http://IP:8080/> 和 <http://IP:8080/axis2> 進行測試（其中的 IP 位址需要替換成 DMSync Server 所在的主機 IP 位址）。

DMSync 可以通過直接執行 `DMSyncAndroidClient1.01.jar` 進行同步，也可以通過在程式中調用 DMSync Client 的 API 介面進行同步。第二種方法可以控制同步的流程，出錯時也可以獲得更多的資訊。兩者都可以使用以下兩個參數。

(1).SQLite DB Path

表示 RDB 在某種平臺上的完整路徑，並且 RDB 已經在 DMSync Server 中註冊過，如

```
D:\sqlite\rdb(Windows),
\data\data/syscom.dmsync.androidsamples/databases/rdb(Android)
```

(2).DMSync ConnectString

在 TELEBOOK 例子中，使用的 `ConnecStr` 為：

```
dmsync://192.168.1.126:8080/connect?pub=TELEBOOKPUB
```

因為此 `ConnectStr` 中沒有指定用戶名，所以在執行同步時，DMSync 會依據 `PUBNAME` 來搜索 `RSUB` 的資訊，使用 `RSUB` 中提供的用戶名和密碼。

執行同步時，DMSync 會根據 `ConnectionString` 的 `Pubname` 來查詢指定 RDB 中的 `RSUB` 資訊，從而連接 DMSync Server 并根據 `Publication` 和 `Objection` 完成資料同步。

在 Android 程式中使用 DMSync 同步

將 DMSync 提供的 `DMSyncAndroidClient.jar` 以 `Refereced Library` 的形式添加到

Android Project TELEBOOK 中，導入其中的 OperationManager 資料，然後創建一個物件，使用該物件調用前面提及的四個介面。

```
import syscom.dmsync.client.operation.OperationManager;
```

```
OperationManager oper = new OperationManager();
```

在 Android 中使用 SQLite 時，創建的資料庫檔一般會放在/data/data/pacakge's name/database 下，不過通過 android SQLiteDatabase 提供的方法獲得路徑。

```
RDBPath : android.database.sqlite.SQLiteDatabase.getPath()
```

```
ConnectionString : dmsync://192.168.1.126:8080/connect?pub=TELEBOOKPUB
```

```
int rc=-100;
rc=oper.init(rdbpath);
if(rc==0){
    rc=oper.connect(ConnectionString);
    if(rc==0)
    {
        rc=oper.sync();

        rc=oper.disconnect();

    }
}
}
```

通過執行以上程式，DMSync 就會把表中的資料同步到另一端資料庫。在後面 DMSync 演示章節中，會通過 DMSync TELEBOOK SAMPLE 來詳細演示同步的過程。

使用 DMSync Client Jar 程式進行資料同步

在 Console 裏切換到 DMSyncAndroidClient1.01.jar 所在目錄，然後執行命令進行資料同步。

```
java -jar DMSyncAndroidClient1.01.jar "RDB Path" "ConnectionString"
```

6 DMSync 同步出錯處理

DMSync 提供的四個 API 介面都會有返回值，執行成功返回 0，失敗則返回錯誤代碼，根據錯誤代碼可以獲得更多的錯誤資訊，詳細內容請參考 `DMSyncError.doc` 文件。

7 同步資料衝突

衝突情況往往發生在一對多關係的資料庫中，如多個 RDB 對應一個 MDB，這時主鍵衝突就不可避免，DMSync 要求為每一張資料表創建主鍵以確保資料的完整性。在一般的應用程式中，如果插入資料時遇到衝突，會給應用程式和資料庫帶來難以預料的後果，而 DMSync 的資料衝突解決策略就是為了解決這些衝突問題而設計的。根據衝突發生的條件來制定相應的解決策略，有以下三種情況：

1) 向資料庫端嘗試同步一筆 Insert 更改時，發現此行已經插入，表示檢測到衝突，則執行對應的 UserCommand。

2) 向資料庫端嘗試同步一筆 Delete 更改時，發現此行已經刪除，表示檢測到衝突，但這種衝突將被忽略，不會對這些資料進行恢復或確認。

3) 向資料庫端嘗試同步一筆 UPDATE 更改時，發送的資料中不僅包含該行的新值（後映射），而且還包含舊值的副本（前映射）。如果前映射與 MDB 中的當前值不匹配，表示檢測到了衝突，此時 DMSync 會根據用戶 OBJ 的 UP_OVR_CMD 或 DN_OVR_CMD 解決衝突，其中包括兩種解決策略，分別是 Server 和 ModifyTime。

下面對相關概念進行介紹說明：

OBJOPTION

OBJOPTION 是存儲在 OBJ 中的一個串，用於設置 OBJ 所採用的衝突解決策略。OBJOPTION 由五個數字組成，目前只需設置第一位數位，0 表示 Server 方式，1 表示 ModifyTime 方式。

Server 方式

以 MDB 資料為主，當 DMSync 進行 UPLOAD 時，如果在 MDB 上發生衝突，DMSync 會把正常的資料插入到 MDB 資料表中，衝突記錄為 do nothing 即不作任何處理。當進行 DOWNLOAD 時，如果在 RDB 上發生衝突，DMSync 將執行 DN_OVR_CMD 命令。

當設置了 PUBOPTION 只進行 UPLOAD 或 DOWNLOAD 單一動作時，衝突發生後以 Server 為主的衝突解決策略將非常有效。

以 TELEBOOK 為例，在同步之前，RDB 端的 TELEBOOK 和 MDB 端的 TELEBOOK 分別插入兩條記錄，如果主鍵值相同，同步時就會發生衝突，要解決此衝突需要更改 DMSYNC_SYNC_OBJ 的值，說明如下：

UP_OVR_CMD

忽略，即仍為空。

DN_OVR_CMD

與 DN_UPD_CMD 類似，但忽略了前映射 OLD 值的比對。

```
UPDATE TELEBOOK SET
  _ID=${NEW._ID},NAME=${NEW.NAME},PHONE_NO=${NEW.PHONE_NUMBER}
WHERE _ID=${NEW._ID}
```

如此以來，DMSync 執行 DOWNLOAD 時，RDB 端資料衝突記錄 MDB 端相應的資料

所替換（_ID 一樣的）。此處的 UserCommand 與 DMSync 系統中 UPDATE 類型的 UserCommand 比較類似，只不過在 Where 子句中去掉了前映射值的比較。當衝突發生時，會採用 Server 的方式來解決衝突，UPLOAD 時可以執行任何 UserCommand，DOWNLOAD 時執行此處的 UserCommand 將 RDB 資料表中發生主鍵衝突的記錄替換為 MDB 中的資料。

ModifyTime 方式

這種方式要求為資料表增加額外一個欄位 **TIMESTAMP** 來存儲每筆記錄的修改時間。當衝突發生時，比對兩端資料庫衝突資料的修改時間來選擇要保留的記錄。

DMSync 約定的 **ModifyTime** 方式以修改時間較早的資料為主來解決衝突，因此 UP_OVR_CMD 和 DN_OVR_CMD 的條件判斷使用“>”，表示時間較早的資料優先，也就是 Where 子句是已經確定的。

以 TELEBOOK 為例，首先為 RDB TELEBOOK 和 MDB TELEBOOK 增加欄位：

```
ALTER TABLE TELEBOOK ADD COLUMN LAST_MODIFY_TIME TIMESTAMP DEFAULT NOW();
```

編寫 **ModifyTime** 方式解決衝突 UserCommand 時，仍然可以通過 PHONE_NO 和 PHONE_NUMBER 來區分 RDB TELEBOOK 和 MDB TELEBOOK。

UP_OVR_CMD

根據提到的 DMSync 約定，UP_OVR_CMD 應滿足條件：

```
MDB.TELEBOOK.LAST_MODIFY_TIME>RDB.TELEBOOK.LAST_MODIFY_TIME
```

那麼 UP_OVR_CMD

```
UPDATE TELEBOOK SET
  _ID=${NEW._ID},NAME=${NEW.NAME},PHONE_NUMBER=${NEW.PHONE_NO},
  LAST_MODIFIED=${NEW.LAST_MODIFIED}
WHERE _ID=${NEW._ID} AND LAST_MODIFIED>${NEW.LAST_MODIFIED} '
```

DN_OVR_CMD

根據提到的 DMSync 約定，DN_OVR_CMD 應滿足條件：

```
RDB.TELEBOOK.LAST_MODIFY_TIME>MDB.TELEBOOK.LAST_MODIFY_TIME。
```

那麼 DN_OVR_CMD

```
'UPDATE TELEBOOK SET
  _ID=${NEW._ID},NAME=${NEW.NAME},PHONE_NO=${NEW.PHONE_NUMBER},LAST_MODIFIED=${NEW.LAST_MODIFIED}
WHERE _ID=${NEW._ID} AND LAST_MODIFIED>${NEW.LAST_MODIFIED} '
```

從上面兩個 UserCommand 可以看出，Where 條件子句表示的是以更改時間較早的資料為準。以 Telebook 為例，當在 RDB 端發生資料衝突時，如果用戶使用 **ModifyTime** 方式處理衝突就會執行此 UserCommand，如果 RDB 端的資料表衝突記錄的修改時間比 MDB 端的晚，才會執行此 UserCommand，並將 RDB 端的資料更新成 MDB 資料。相反，如果 RDB 端的資料表中，衝突記錄的修改時間比 MDB 端的早，DMSync 也會執行此 UserCommand，但是很顯然，RDB 端並沒有發生任何資料的更新。

當衝突發生在 MDB 端時，也屬於同樣的策略。**ModifyTime** 方式需要執行 UPLOAD 和 DOWNLOAD 這兩個過程才能最終將兩端資料保持一致。

8 不同類型的映射

本手冊將 SQLite 和 DBMaker 分別作為 RDB 和 MDB。DMSync 將需要同步的資料封裝成 xml 資料包在網路上傳輸，因此 RDB 和 MDB 的類型轉換是通過 XML 來完成的。當 DMSync 執行 UPLOAD 動作時，首先將 SQLite RDB 的資料進行 XML 打包，也就是轉換成 XML 類型。當 XML 資料包到達 DMSync Server 時，再對 XML 資料進行解析，也就是把 XML 類型轉換成 DBMaker 的資料類型，接下來執行 DOWNLOAD。

SQLite 是弱類型的資料庫，擁有五種基本類型，與 XML 資料類型的轉換列表如下：

映射表: (Sqlite type<-> XML type)

Sqlite Storage Type	XML Type
Blob	BLOB
Text	CLOB
Integer	BIGINT
Real	DOUBLE
null	NULL

DBMaker 是一個強類型的資料庫，它與 xml 資料類型的轉換列表如下：

XML 類型	DBMaker 類型
Blob	LONG VARBINARY
CLOB	CHAR
	VARCHAR
	CLOB
	NCHAR
	NVARCHAR
	BLOB
	NCLOB
	TIMESTAMP
BIGINT	TIME
	DATE
	SMALLINT
	DECIMAL
	DOUBLE
	FLOAT
	INTEGER
	LONG VARCHAR
	NULL
	REAL
VARCHAR	
CHAR	

	INT
	BIGINT
	SERIAL
	BIGSERIAL
DOUBLE	FLOAT
	BIGINT
	SERIAL
	BIGSERIAL
	CHAR
	DECIMAL
	FLOAT
	INTEGER
	LONG VARCHAR
	NULL
	SMALLINT
	VARCHAR
	DOUBLE
NULL	All DBMaker type can casting

因此，用戶在設計 DMSync 同步時，應嚴格遵守所支援的類型列表。

9 DMSync 性能調優

9.1 PackageSize

DMSync 通過 JDBC 獲取資料庫中的資料，然後將這些資料封裝成 xml 包，再通過 Webservice 在網路上進行傳輸。這些在網路上傳輸的資料包大小是可以設置的，DMSync 默認將 UPLOAD 和 DOWNLOAD 的 package size 設置成了 4096 Bytes，也就是 4KB。DMSync 將這些同步的資料進行封包時，如果其大小小於 4KB 就會直接傳輸，如果大於 4KB 就會將其封裝成多個包，並且每個包都小於 4KB。

DMSync 允許用戶根據實際的網路環境來設置資料包的大小，包括 UPLOAD 和 DOWNLOAD 的資料包。

UPLOAD PACKAGE SIZE

上傳的包大小是在 RDB_SYNC_SYS 表中進行設置的，這部分內容已經在 DMSync 同步部署章節中進行了描述。

```
Insert into RDB_sync_sys values('uploadpacketsize','4096');
```

用戶可以使用 UPDATE 對此記錄進行修改，從而完成對包大小的設置。

DOWNLOAD PACKAGE SIZE

下載時則需要在 dmsync.xml 中進行設置，在配置檔中可找到用來設置資料包大小的節點，如下：

```
<download>
  <!-- <MaxDownloadItemSize></MaxDownloadItemSize> -->
</download>
```

上例中用來設置下載資料包大小的節點被注釋掉了，所以下載時資料包大小會採用預設值 4KB。如果用戶要對其進行設置，只需要刪除 MaxDownloadItemSize 的注釋並填寫相應的數值即可（單位是 Byte）。

10 DMSync 演示

本章將介紹如何使用 DMSync 提供的 TELEBOOK SAMPLE 來演示 Android DMSync 的同步。

10.1 運行環境

DMSync Server

Jetty & Axis2 Server : dmsync://192.168.1.126:8080

DMSync DB

DB_SVADR:192.168.1.126

PORT: 3307

LCODE:10 (UTF-8)

MDB

DB_SVADR:192.168.1.126

PORT: 3308

LCODE:10 (UTF-8)

10.2 初始化 DMSync DB

1) 創建 DMSyncDB，其 DBMaker 的配置檔 dmconfig.ini 為：

```
[DMSync]
DB_DBDIR = D:\release\dmsync\workspace\DMSync
DB_FODIR = D:\release\dmsync\workspace\DMSync\fo
DB_UsrFo = 1
DB_UsrID = SYSADM
DB_UMODE = 1
DB_FODIR = D:\DMSync\fo
DB_CTIMO = 5
DB_SVADR = 127.0.0.1
DB_PTNUM = 3307
DB_BMODE = 0
```

2) 執行腳本 TELEBOOK_DMSYNC.sql。

```
dmSQL>run TELEBOOK_DMSYNC.sql;
```

3) 啓動 DMSync DB。

10.3 初始化 MDB

1) 創建 MDB，其 DBMaker 的配置檔 dmconfig.ini 爲：

```
[MDB]
DB_DBDIR = D:\release\dmsync\workspace\MDB
DB_SVADR = 127.0.0.1
DB_PTNUM = 3308
DB_USRID = SYSADM
DB_DBFIL = MDB.SDB
DB_BBFIL = MDB.SBB
DB_USRDB = MDB.DB 200
DB_USRBB = MDB.BB 3
DB_JNFIL = MDB1.JNL MDB2.JNL MDB3.JNL MDB4.JNL
DB_JNLSZ = 1000
DB_LCODE = 3
DB_IDCAP = 1
DB_BFRSZ = 32
DB_FODIR = D:\release\dmsync\workspace\MDB\fo
DB_USRFO = 1
DB_UMODE = 1
```

2) 執行腳本 TELEBOOK_MDB.sql。

```
dmSQL>run TELEBOOK_MDB.sql;
```

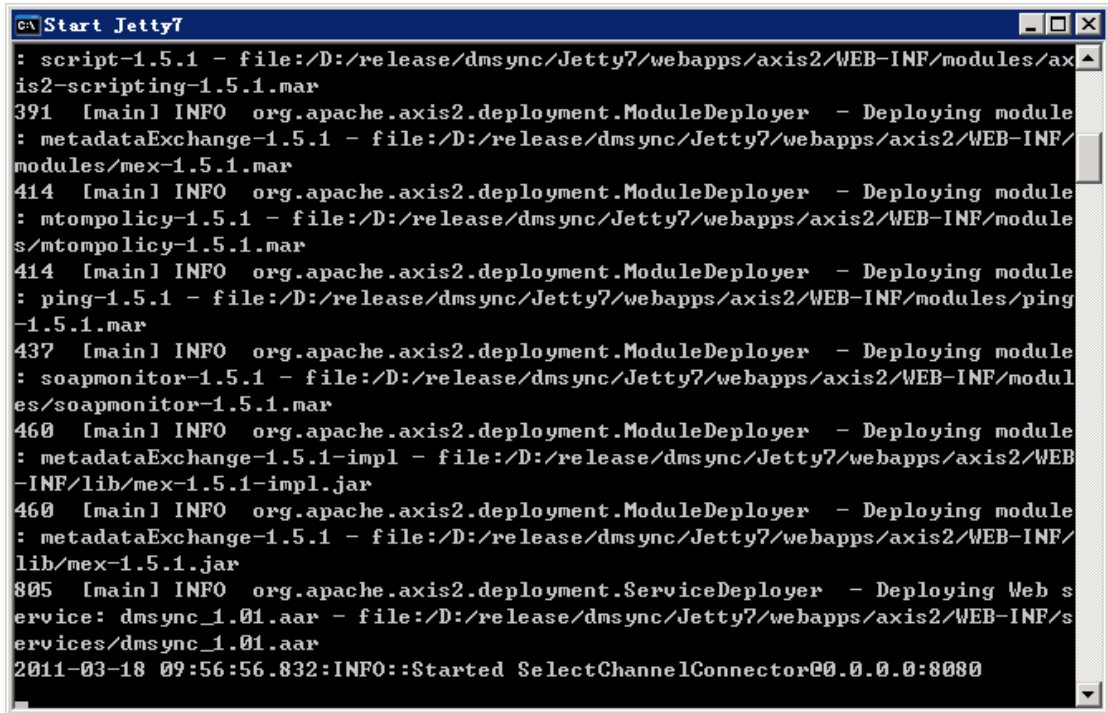
3) 啓動 MDB。

10.4 啓動 Jetty

首先進入 cmd line 模式並切換至 Jetty 目錄，然後在命令行中輸入如下命令啓動 Jetty。

```
Jetty7>java -jar start.jar
```

或者通過“資源管理器”直接點擊 start.bat 來啓動 Jetty，如圖所示：



```
C:\> Start Jetty7
: script-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/modules/axis2-scripting-1.5.1.mar
391 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: metadataExchange-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/modules/mex-1.5.1.mar
414 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: mtompolicy-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/modules/mtompolicy-1.5.1.mar
414 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: ping-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/modules/ping-1.5.1.mar
437 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: soapmonitor-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/modules/soapmonitor-1.5.1.mar
460 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: metadataExchange-1.5.1-impl - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/lib/mex-1.5.1-impl.jar
460 [main] INFO org.apache.axis2.deployment.ModuleDeployer - Deploying module
: metadataExchange-1.5.1 - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/lib/mex-1.5.1.jar
805 [main] INFO org.apache.axis2.deployment.ServiceDeployer - Deploying Web service: dmsync_1.01.aar - file:/D:/release/dmsync/Jetty7/webapps/axis2/WEB-INF/services/dmsync_1.01.aar
2011-03-18 09:56:56.832:INFO::Started SelectChannelConnector0.0.0.0:8080
```

可以在視窗末端看到在 8080 埠啟動成功的資訊。

10.5 在 Android 上初始化 RDB 并執行同步

手冊提供的 TELEBOOK Sample 是一個 Eclipse 下的 Android Project，可以通過 Eclipse 將其直接導入。

TELEBOOK Android Project 下包含了初始化 RDB 的 SQL 語句，放在以下目錄：

TELEBOOK/value/rdbinit.xml

啟動 TELEBOOK

啟動 TELEBOOK APP 之後可以看到下圖：

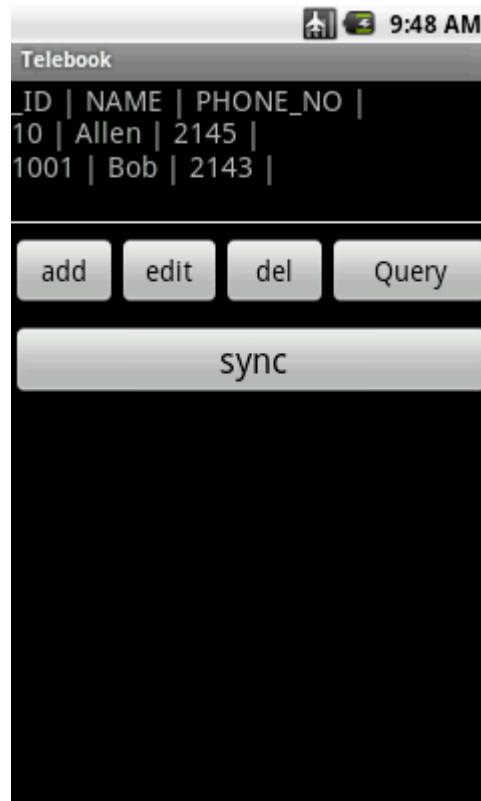


圖 9.1

圖中有五個 **BUTTON**，可以簡單地從名稱上判斷出各個按鈕的功能。

Add

Add 可以添加預設記錄。

EDIT

EDIT 可以修改已經添加的預設記錄。

DEL

EDIT 可以刪除修改過的預設記錄。

Query

刷新資料表，獲取其他程式對資料表的更改。

Sync

執行同步，將 RDB TELEBOOK 更改同步到 MDB。

同步 INSERT 更改

點擊 TELEBOOK 上的 Add 按鈕後再點擊 Sync，TELEBOOK APP 會提示已經同步成功的資訊，提示資訊中還包含了 DMSync Server 的位址。

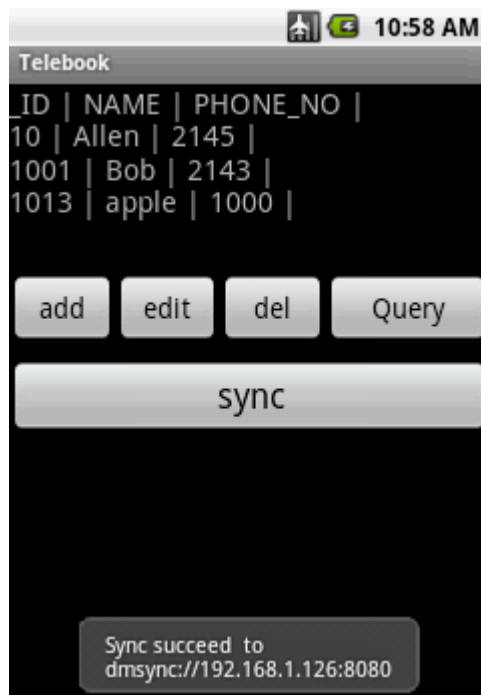


圖 9.2

此時查看 MDB TELEBOOK，可以看到 INSERT 插入的資料已經同步到了 MDB。

_ID	NAME	PHONE_N...	AGE	COMPANY	DEPARTM...	JOINDATE	EMAIL
10	Allen	2145	...				
1001	Bob	2143	...				
1013	apple	1000	...				

圖 9.3

同步 UPDATE 更改

點擊 TELEBOOK APP 上的 edit 按鈕，可以看到剛插入的資料已經變更，然後點擊 Sync 按鈕會看到同步成功的資訊。

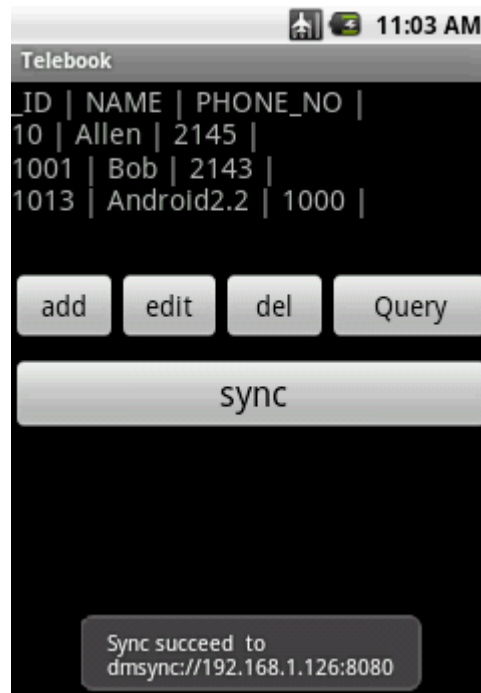


圖 9.4

上圖中_ID 為 1013 記錄的 NAME 值已經從 Apple 變更至 Android，同步後再查看 MDB 的資料，可以看到 UPDATE 的更改已經同步到了 MDB。

选择的记录 3。(1 - 3)

_ID	NAME	PHONE_NU...	AGE	COMPANY	DEPARTM...	JOINDATE	EMAIL
10	Allen	2145
1001	Bob	2143
1013	Android2.2	1000

圖 9.5

同步 DELETE 更改

為了體現 DELETE 這個動作，用戶可先點擊添加按鈕添加一條記錄，此時可以看到下圖中新添加的 1014 記錄，然後再點擊 DEL 按鈕刪除 1013 記錄。



圖 9.6

從這兩幅圖中可以看到_ID 為 1013 的記錄已經被刪除，。查看 MDB TELEBOOK 可得知 Delete 的更改也同步了。

_ID	NAME	PHONE_N...	AGE	COMPANY	DEPARTM...	JOINDATE	EMAIL
10	Allen	2145	...				
1001	Bob	2143	...				
1014	apple	1000	...				

圖 9.7

10.6 MDB 資料更改的同步

當 MDB 端的資料發生更改時，同樣會同步到 RDB 端。

在 MDB 端的 TELEBOOK 中增加一條含中文的記錄，然後在 TELEBOOK APP 上點擊 Sync 按鈕完成同步。

_ID	NAME	PHONE_N...	AGE	COMPANY	DEPARTM...	JOINDATE	EMAIL
10	Allen	2145	...				
100	张三	4564	22				
1001	Bob	2143	...				
1014	apple	1000	...				

圖 9.8



圖 9.9