**数据库信息同步设计初步方案**

**一、项目背景**

**Rio是一款新型跨设备富信息的数据管理器，在不同终端中对系统内信息、文件提供了同类型多角度的展示方式，方便于用户的检索；对前端应用开发提供了统一稳定的接口，方便与开发者的上层应用开发。现考虑实现当两台同时装有Rio数据管理器的设备在用户操作下开始网络通信时进行数据库信息的自动同步。此同步并不是以往简单的数据库迁移、替换或复制，而是基于每条数据中每个字段的智能更新，尽可能满足用户使用需求，简化用户操作内容，减少信息传递信息，优化服务性能。**

**二、基本原理**

**记录用户对每一条数据信息的操作（CUD），并对已有的每一条数据每一个字段进行监听，记录对于其内容的每一次操作的行为（CUD）及内容，在次基础上在最大满足用户要求情况下进行数据合并更新，并在数据同步完成后记录同步设备信息与时间信息。**

**对于CUD这三种操作行为，建立对应表来存储其具体操作内容，表结构主要内容为操作对象的ID、KEY、VALUE及上一次操作行为的ID；以一系列有序而统一的操作行为构成针对这一数据操作的版本记录；在进行数据同步的同时完善版本记录的内容并通过对版本记录的回溯来完成对数据的版本控制。**

**三、基本规则**

**1.以每条数据中的字段为最小操作单元--数据元，每一条数据都由多个数据元构成。**

**2.对于一台设备上新建一条数据信息无论其来源如何都会认为是一条全新数据；而从其他设备进行数据同步时得到的新数据不算做全新数据。**

**3.两个设备上相同ID的数据对比时，仅一条有操作记录情况下，不算做冲突；当产生冲突时进行数据元对比。**

**4.数据元对比时从有到无或从无到有都不算是冲突；当数据元对比时其中一个的当前版本为另一个的中间版本时也不算是冲突；确认为冲突时提交至用户解决。**

**5.一台设备上对于同一个数据元的每一次操作行为都应有准确的记录，一系列有序的操作行为构成该数据元在此设备上的版本记录。**

**6.多台设备同步时，同一数据元的版本记录也需要同时同步，其数据结构应为有向图结构。**

**7.在同一设备上，其版本记录结构可相对看做为有始有终的有向图；在多设备上来看其版本记录在完美情况下也应为有始有终的有向图。**

**8.要保证数据同步时版本记录的完整性，即使是某一条被放弃的版本记录分支也应该完整的保存下来。**

**9.在版本记录终点选择上，如果用户希望保留两个冲突的记录，所进行的处理方式为选择其一为终点，对另一数据进行新建处理。**

**10.版本记录起点由操作行为C带来，在UD处结束。**

**11.在每条记录中增加is\_delete字段，默认为0；前段页面展示给用户的是该字段为0的所有文件。**

**12.当用户删除文件时，首先执行的是update操作，设置is\_delete为1；文件不会直接进行删除，回收站中展示的是该字段为1的文件信息。**

**13.当用户永久性删除某一文件时，其他所有设备中该文件应同步删除。**

**14.当数据信息与对应文件全部删除是算作真正删除，需要被记录。**

**15.永久删除记录在delete操作表中时，应同时处理该条数据在insert和update操作表中所有的相关记录。**

**16.当对同一个数据元进行多次操作时，每一次操作都需要对数据元所在数据ID、数据元KEY及VALUE进行记录。**

**17.每次同步需记录两台设备间同步信息（如设备名称，增加信息数量，更新信息数量，删除信息数量，同步时间等）**

**18.操作优先：删除>新建>更新**

**19.规则是双向的，尽量满足用户需求，用户也需一定程度遵循设计原理**

**四、逻辑方法**

**1.以表的形式分别记录用户的不同操作行为（CUD）及内容。**

**2.以表的形式记录各台设备间同步信息。**

**2.当用户连接两台都具有Rio的设备时，Rio征得用户同意后进行数据自动同步。**

**3.假定两台设备分别为A与B，首先查询同步信息表判断是否为首次同步：**

**1）如果为首次同步：**

**（1）获取A中删除记录中的所有内容传递给B，在B中取得全部删除操作后进行对比后删除，成功后返回B中删除记录返回给A进行操作。**

**（2）获取A中新增记录中的所有内容传递给B，在B中进行插入操作，成功后返回给A进行新增操作。**

**（3）对比A与B中更新记录对象：**

**a.将A内更新对象中与B中重复部分取出传递给B进行更新操作**

**b.将传递过来的B内更新对象重复部分取出在A中进行更新操作**

**c.将两设备中重复更新操作的每一条数据的数据元进行对比，如对统一数据元进行操作则作为冲突提交至用户解决，如不存在对统一数据元操作则进行合并，并在A中进行更新操作。**

**d.A中更新结束后将合并后交集更新操作返回给B进行更新操作**

**（5）B中更新成功后记录两设备同步信息，同时返回信息给A**

**（6）A收到信息进行同步信息记录，结束**

**2）不是首次同步：**

**根据同步信息表中记录的上次同步信息，选取其后的创建、更新、删除信息进行同样操作**

1. **Nosql还是sql**

数据量级：22万（文件数）\*100（标签数）

并发性：低

扩展性：高

实时性：中

查询性能：高

写性能：低

存储空间：高

内存空间：高

分布式：无

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 数据量级 | 并发性 | 扩展性 | 实时性 | 查询性能 | 写性能 | 存储空间 | 内存空间 | 编程友好 | 分布式 | 嵌入式 |
| Mongodb |  | √ | √ | √ |  | √ |  |  | √ |  |  |
| Sqlite | √ |  |  | √ | √ | √ | √ | √ | √ | √ | √ |

目前倾向使用sqlite。

参考资料：

（1）MongoDB的基本特性与内部构造

http://www.nowamagic.net/librarys/veda/detail/1884

（2）SQL vs NoSQL Database Differences Explained with few Example DB

http://www.thegeekstuff.com/2014/01/sql-vs-nosql-db/

（3）MongoDB vs MySQL

http://blog.csdn.net/li\_yang98/article/details/6198871

（4）NoSQL数据库探讨之一 － 为什么要用非关系数据库？

http://robbin.iteye.com/blog/524977

（5）MongoDB的真正性能

http://linux.cn/thread-10563-1-1.html

（6）MongoDB与内存

http://huoding.com/2011/08/19/107

（7）SQL vs NoSQL：数据库并发写入性能比拼

http://www.tsingpost.com/articles/201401/387.html

（8）入门MongoDB，你需要注意的4个技巧

<http://www.zeuux.com/group/mongodb/bbs/content/37225/>

（9） [MongoDB学习笔记](http://www.cnblogs.com/lipan/archive/2011/03/08/1966463.html)

http://www.cnblogs.com/lipan/archive/2011/03/08/1966463.html

（10）[SQLite性能和限制](http://blog.csdn.net/yuzhouxiang/article/details/7373111)

<http://blog.csdn.net/yuzhouxiang/article/details/7373111>

1. **版本控制针对sql语句还是针对数据**

两种方法都能记录数据库的变化历史，但是两个sql语句序列合并不能得到确定的结果。所以选择对数据进行版本控制，但是需要消耗更多额外的存储空间。

参考资料：

（1）Database Versioning: The ignored aspect of version control.

http://www.nimkar.net/index.php/9-release-management/3-database-versioning-the-ignored-aspect-of-version-control

（2）Database Version Control

<http://techportal.inviqa.com/2011/01/11/database-version-control/>

1. **现有工具？**

Mongodb：LiquiBase MongoMVCC Vermongo

Sqlite: LiquiBase

现有工具主要用于数据库更新、数据库回滚等操作，不支持类似merge的操作。Vermongo似乎支持，但是项目不成熟而且两年期就没更新了。需要自己开发版本控制系统。

参考资料：

（1）Vermongo

https://github.com/thiloplanz/v7files/wiki/Vermongo

（2）representing-revision-data-in-mongodb

http://software.danielwatrous.com/representing-revision-data-in-mongodb/

（3）mongomvcc

https://github.com/igd-geo/mongomvcc

https://github.com/igd-geo/mongomvcc/wiki

（4）mongomvcc 0.5.0 API

http://igd-geo.github.io/mongomvcc/api/current/

（5）Liquibase安装及使用

http://www.penglig.com/post-267.html

（6）让开发自动化: 实现自动化数据库迁移

http://www.ibm.com/developerworks/cn/java/j-ap08058/index.html

（7）Liquibase documentation

https://github.com/liquibase/liquibase

http://www.liquibase.org/documentation/index.html