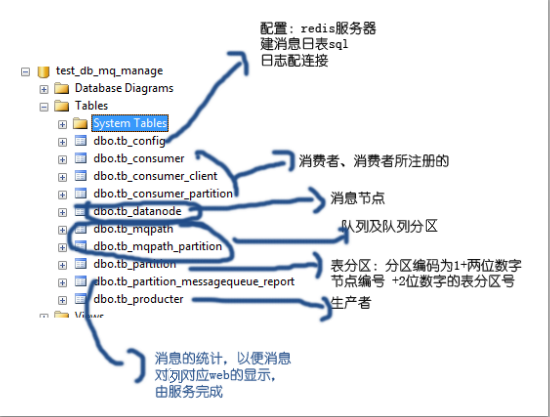
BusinessMessageQueue实现分析

# 概要

BusinessMq实现对消息的转发、存储、通知，具有高可靠性，因为消息持久化在数据库内，所以不会丢失且在发生故障停机时，可以恢复到原状态继续处理消息。

当消息服务器设置有多个节点，且消息队列在每个节点上都有分区时，当一个节点发生故障，消息会自动负载到无故障的节点，这种故障隔离机制使得Business的可靠性大大增强。

# 消息管理库分析



表分区的的partitionid信息：1+{两位数字节点}+{两位数字表分区序号}

队列消息日表的表名规则：tb\_messagequeue\_{两位表分区号}\_{yyMMdd}

队列消息日表的消息id为自增：初始值为{表分区的partitionid}+{六位数字的日表日期yyMMdd}00000001

# 消息队列相关的必要服务分析

* 建日表服务：保证消息节点有预3天的日表
* 消息统计服务：对当前生产的、消费的、未消费的消息进行统计，以便web的实时查看，因为涉及到跨库，所以直接查询并不方便。此任务的每次执行时间相对较短，保证管理web查看当前消费者状态信息时为当前状态信息。

关于服务的说明：

该案例中的服务运行在一个任务平台上，该平台能对任务进行管理和状态反馈，由此处是队列的实现分析，因此不讨论任务平台功能，只要能记这些服务按时自动运行即可。

# 消息队列对应管理Web

## 网站截图



## 网站功能介绍

1. 队列的管理配置，分区管理
2. 队列运行状态的查看
3. 异常错误的查看分析，当某个队列出错误，可以通过配置这个队列为调试队列来实时输出错误，方便分析问题。
4. 消费者管理页可以查看当前队列的消费者，从这个页面可以查到队列消息状态，消费情况，速度，如果消费的速度过慢，可以考虑增加分区，加大并行处理线程数。

# 消息的生产者

一个队列可有多个生产者，在新建生产者实例并Open的时候，会新增一个生产者并删除没有心跳的生产者

## 打开时会进行一些操作

* 检查mqpath有效性
* 取得当前mqpath的所有表分区
* 得到mqpath所有在的所有表节点
* 删除没心跳的生产者
* 新增当前生产者
* 注册心跳线程，redis内部通信通道监控线程。

*因为每次打开的都会进行新增操作，所以生产者实例应为静态共用。*

## 同步生产者服务器时间：Open方法、生产者心跳处理程序

## 生产者的重启条件：（其一）

1. 心跳监控到队列配置有修改
2. 收到reids内部通信重启信息
3. 有表分区添加信息错误超过1分钟

## 消息的发送：将消息发送到队列表分区

当有多个表分区时，有个负载均衡算法，

## 默认的负载均衡算法：

系统唯护一个数字，这字数据会在每次调用这个算法时递增，根据这个数字，系统循环的在当前有效的分区中取一个分区。

如果插入这个表分区插入数据失败，系统会将这个表分区所在的节点的生的表分区都移出到错误天分区并记录错误时间以并重启。

当重试三次仍然失败，则插入消息失败。

生产者重启（系统叫更新生产者缓存信息)时会重取得到表分区、节点和清理上次错误的分区。

# 消息的消费者

## 配置说明

最大接收线程数及最大并行分区数

## 消费者注册时：

* 删除上次的消费者（根据消费者ID和临时消费者ID）
* 根据消费者名称ID得到原有或新建消费者客户端(Customer\_client)
* 注册消费者：先删除没有心跳的消费者（根据当前的客户端ID和过期心跳），
* 得到消费者客户端已ID所注册过的所有分区
* 得到已注册分区与当前欲注册分区的冲突表分区

如果有冲突则注册异常，注册失败。

无则成功，然后添加消费者。并得到消费者信息

* 根据消息队列名得到消息队列信息
* 添加消息队列表分区的注册信息：
  + 按分区一个个添加{如果队列没有该index的分区，继续}
  + 消费者分区表中如果已有注册则更新，没有则添加，并初始化上次消费ID为初始值（取mq最大值或默认初始值）
* 初始化到当前消费者的表分区中（内存）
* 初始化到当前消费者的节点中（内存）
* 初始化消费者服务器时间（每次取队列数据到内存时还会更新，已便日表换时的换表）
* 注册消息处理的回调：
* 每个队列分区的初始化和redis的mq信息通道监听和一个定时取信息任务
* 消费者的心跳（包括队列配制信息的修改重载和心跳记录和redis系统内部消息）

## 重启消费者的条件

队列信息修改（最后修改时间并化)、收到redis的重启内部信息

## 初始化消费者服务器时间：

消费者启动、每次取消息

## 消费者的内部细节

* 调用方面：

处理消息的触发机制有两个来源：一是生产者生产一条消息时会给redis对应的那个队列通道发送一条消息，表示有新消息需要处理

二是有一个定时取信息的线程：每隔Consumer\_ReceiveMessageQuque\_Every\_Sleep\_Time的时间会去取消息，

也就是说如果redis挂了，那么消息的响应处理时间会有所延时。

当监听redis消息的线程收到新消息提醒时，并不是立即读数据库取消息，而是等待一小段时间再取，这在生产者的并发高时尤为有用。

* 处理方面：

当得知有新消息要处理时，会单线程的去消息队列的每个分区表中取N条新数据（前一次的ID会记录下来，按大于这个ID来取），

然后按处理的最大线程数并行处理：最大线程最大为分区数，最小为1，取过在大或过小会自动调整。

处理时如果出错或未标记处理成功，会记录这个分区表，下次取数据时会跳过这个分区，这也是为了保存单个分区的消息是有序处理的。处理一条消息会更新这个分区的最后ID，已便重新取数据或重启，保证不会遗漏消息，处理消息时会过滤第二天的数据和已迁移的数据，

* 取数据时有个地方值得注意：

当从最后消息ID所在的日表里取不到消息时，会去检查是否是日期的转化点，如果是，则会将当前处理的最后消息ID设为第二天的初始值，但不更新库里的记录（因为这个ID不实际存在，所以如果在这个点将消费者停止，重启时还会做这个过程）

修改时间:10/28/2015 4:58:53 PM

By 徐品