包名:dbcache

为了简化服务升级中对数据的批量入库。特把相关功能集中到本包中。先缓存需要入库的相关数据，当达到一定数量后，集中一次入库。

func NewCache()

声明数据库批量操作对象。需要传入数据库操作对象、表名和字段信息。字段信息可传入表结构体、字段列表map或者表中的自增ID字段名。根据传入的字段信息类型对表字段进行初始化。

func (db \*DBCache)Start()

批量入库自动运行函数。对缓存的数据进行自动入库。当缓存数据达到一定数量或者超过一段时间没有新数据，则执行入库操作。

**func** (db\*DBCache) AppendData(js **interface**{})

数据缓存函数，外部调用时，只需要通过此函数传入数据到达内部缓存。在Start()函数中会针对数据进行处理。

**func** (this \*DBCache) insert\_data(data []**interface**{})

数据写入函数。目前只支持DUPLICATE操作。需要表中有唯一索引。

包使用注意事项

固定表名使用DBCache结构体。而对于按月分表等需要使用DBCacheMonth结构体。DBCacheMonth结构体在声明是只需要传入数据库连接对象和字段，不需要传入表名。在传入数据的时候，不但要传入数据，还需要把插入的表也一并传入。

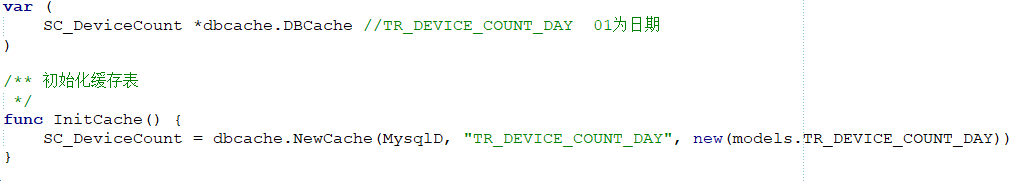
当传入的表名为新表时，会创建对应的DBCache对象用于处理新表的数据，并把该表的对象写入相关列表中，以便下次写入数据时直接使用。

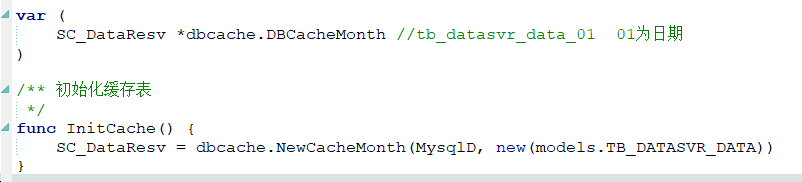
初始化表的时候，要注意只声明数据写入或者更新的字段名，其他的字段不用声明。如ID等字段，声明后程序也会对其进行写入/更新操作，从而导致字段数据被覆盖或者清空为NULL。

使用步骤

1. 初始化对象

固定表名的使用DBCache结构,按月分表的使用DBCacheMonth结构。





1. 数据写入

可通过声明表结构体对象，赋值后写入。也可创建map，通过map进行数据传递。

固定表写入：

js := new(models.TR\_DEVICE\_COUNT\_DAY)

SC\_DeviceCount.AppendData(js)

月表写入：

tabName := fmt.Sprintf("tb\_datasvr\_data\_%02d", time.Now().Day())

SC\_DataResv.AppendData(tabName, jsD)