数据自动分组（grouping）

- 分区（partition）也是一种数据分组的方式
  - 提高并发性（concurrency）和并行性（parallelism）
  - 从而增强系统架构的可伸缩性（scalable）

循环分区

- 循环分区：不受数据影响的内部机制
  - 分区定义为各个磁盘的存储区域
  - 可以看作是随意散布数据的机制
  - 保持更改带来的磁盘I/O操作的平衡

数据驱动分区

- 根据一个或多个字段中的值来定义分区
  - 一般叫分区视图（partitioned view），而MYSQL称为（merge table）
- 分区的实现方式
  - 哈希分区（Hash-partitioning）
  - 范围分区（Range-partitioning）
  - 列表分区（List-partitioning）

分区是把双刃剑

- 分区能解决并发问题吗?
- 又回到了IOT类似的问题： “冲突”
  - A. 通过分区键将数据聚集，利于高速检索;
  - B. 对并发执行的更改操作，分散的数据可以避免访问过于集中的问题
- So，A or B……完全取决于您的需求

分区与数据分布

- 表非常大，且希望避免并发写入数据的冲突就一定要用分区吗?
- 例如客户订单明细表……
- 对分区表进行查询，当数据按分区键均匀分布时，收益最大

数据分区的最佳方法

- 整体改善业务处理的操作，才是选择非缺省的存储选项的目标
- 更新分区键会引起移动数据，似乎应该避免这么做
  - 例如实现服务队列，类型（T1...Tn）状态（{W|P|D}）
  - 按请求类型分区：进程的等待降低
  - 按状态分区：轮询的开销降低
  - 取决于：服务器进程的数量、轮询频率、数据的相对流量、各类型请求的处理时间、已完成请求的移除频率
- 对表分区有很多方法，显而易见的分区未必有效，一定要整体考虑

Holy Simplicity

- 除了堆文件之外的任何存储方法，都会带来复杂性
- 除了单库单表之外任何的存储方式，都会带来复杂性
- 选错存储方式会带来大幅度的性能降低
- 总结
  - A. 测试，测试，测试
  - B. 设计是最重要的
  - C. 任何设计都有时效性

End

下一模块，我们讲数据库模式设计的一些问题。