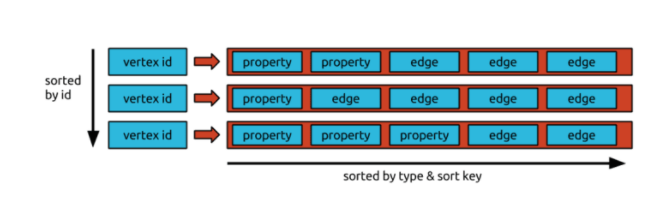
# **图分区**

·JanusGraph以邻接表形式存储图数据，顶点的分配到机器上的方法就决定了图的分区。



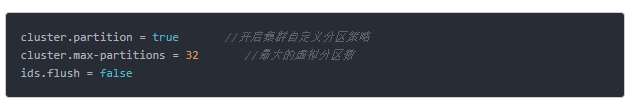
# **随机分区策略（默认策略）**

· 随机安排顶点到所有机器上。缺点：查询效率慢，因为存在大量的跨实例的通信。

# **显示分区**

·具有强关联性和经常访问的子图存储在相同的机器上，这样可以减少跨机器的通信成本。

配置参数



· max-partitions：最大虚拟分区数，建议配置为存储数据个数的两倍。

· 精确分区只有支持key排序的存储后端

·Hbase：支持图自定义分区

·Cassandra：需要配置ByteOrderedPartitioner来致辞图分区

· 在图分区下有edge cut和vertex cut两方面可以单独控制。

**Edge Cut(默认)**

目的：对于频繁遍历的边，应该减少cut edge的存在，从而减少跨设备间的通信，提高查询效率。即把进行遍历的相邻顶点放在相同的分区，减少通信消耗。

为顶点确定分区：JanusGraph通过配置好的 placement strategy来控制vertex-to-partition的分配。

默认策略：在相同事务中创建的顶点分配在相同分区上。

缺点：如果在一个事务中加载大量数据，会导致分配不平衡。

定制分配策略：实现IDPlacementStragegy接口，并在通过配置文件的ids.placement项进行注册。  
**Vertex Cut**

Vertex Cut：顶点切割，即把一个顶点进行切割，把一个顶点的邻接表分成多个子邻接表存储在图中各个分区上。

目的：一个拥有大量边的顶点，在加载或者访问时会造成热点问题。Vertex Cut通过分散压力到集群中所有实例从而缓解单顶点产生的负载。

方法：JanusGraph通过label来切割顶点，通过定义vertex label成partition，那么具有该label的所有顶点将被分配在集群中所有机器上。

案例：对于product和user顶点，product顶点应该被定义为partition，因为用户和商品有购买记录（edge），热销商品就会产生大量的购买记录，从而会造成热点问题。

