	toVertexType).property("tsU S",0)	

滚动Rotate功能描述

ByteGraph支持数据滚动删除(rotate)功能，该功能针对同一个点的出边进行设置，允许按照配置进行定长FIFO滚动删除。

由于历史迭代遗留，目前该功能有两种模式，后续新集群以V2为准：

	Rotate Part（已废弃）	精准Rotate（V2）
生效范围	只针对点的出边	只针对点的出边
触发条件	配置max_part、max_edge_per_part 以part维度进行滚动，在split时发现part数达到阈值，则会删除最早的part	配置max_edges 当点的出边达到阈值后，再插入一定数量的边就会删除最早相同数量的边
注意事项	1. max_part=1时可以保证精准数量的滚动，max_part>1时，由于每个part数量不定，不能保证精确的滚动 2. 暂不支持开启边上索引的场景（包括反向、双向、属性索引）	1. 暂不支持开启边上索引的场景（包括反向、双向、属性索引）

注意事项	1. 插入边时一定要设置tsUs属性（微秒单位） 2. 可以通过调整tsUs来实现每条边不同的过期时间 3. 1.0只实现了写时更新，读时无保证（可能读到过期的边数据，禁止基于精准ttl设计业务逻辑），2.0已修复这个问题，过期的边数据能正常过滤（点数据待修复），上层业务无感知 4. 暂不支持开启边上索引的场景（属性索引） 5. 由于实现原因，TTL过期的数据并不保证过期后一定删除： 处于事务提交前中间状态的数据（数据量级很小） a. 每个起点/终点相关的边如果长时间没有写入发生，最多会残留2k条过期数据，再出发一次写入这些过期数据可以清理 i. 写入一条肯定过期的数据： `g.addE(edgeType).from(fromVertexID, fromVertexType).to(toVertexID, toVertexType).property("tsUS",0)`	1. 1.0 由于底层点的邻接边是一批一批存储的（按tsUs聚合），过期也是按一批一批过期的，每批边根据最后一次更新的时间+ttl过期（所以可能读、写到本应过期数据） 2. 暂不支持开启边上索引的场景（包括反向、双向、属性索引）

滚动Rotate功能描述

ByteGraph支持数据滚动删除(rotate)功能，该功能针对同一个点的出边进行设置，允许按照配置进行定长FIFO滚动删除。

由于历史迭代遗留，目前该功能有两种模式，后续新集群以V2为准：

	Rotate Part（已废弃）	精准Rotate（V2）
生效范围	只针对点的出边	只针对点的出边
触发条件	配置max_part、max_edge_per_part 以part维度进行滚动，在split时发现part数达到阈值，则会删除最早的part	配置max_edges 当点的出边达到阈值后，再插入一定数量的边就会删除最早相同数量的边

数据自动过期TTL&滚动Rotate功能 Time to live & Rotate

owner： @张祯杰

State : reviewing

TTL功能描述

Bytegraph支持 数据自动过期(ttl)功能，该功能可以使得存储进入bytegraph的点、边数据在满足触发条件时，自动过期删除。

具体有两种配置模式，相应的功能以及过期逻辑如下表：

	根据边上tsUs属性计算过期时间模式	全局固定过期时间 模式【已废弃，不再新增】
生效范围	点、边 都有效	点、边 都有效
触发条件	边：系统当前时间 - 边tsUs属性值 > ttl (边tsUs属性值用户在addE时设置的) 点：与“全局固定过期时间 模式”一致	边：系统当前时间 - 边的插入时间 > ttl 点：系统当前时间 - 点的最后一次变更时间 > ttl
注意事项	1. 插入边时一定要设置tsUs属性（微秒单位） 2. 可以通过调整tsUs来实现每条边不同的过期时间 3. 1.0只实现了写时更新，读时无保证（可能读到过期的边数据，禁止基于精准ttl设计业务逻辑），2.0已修复这个问题，过期的边数据能正常过滤（点数据待修复），上层业务无感知 4. 暂不支持开启边上索引的场景（属性索引） 5. 由于实现原因，TTL过期的数据并不保证过期后一定删除： 处于事务提交前中间状态的数据（数据量级很小） a. 每个起点/终点相关的边如果长时间没有写入发生，最多会残留2k条过期数据，	1. 1.0 由于底层点的邻接边是一批一批存储的（按tsUs聚合），过期也是按一批一批过期的，每批边根据最后一次更新的时间+ttl过期（所以可能读、写到本应过期数据） 2. 暂不支持开启边上索引的场景（包括反向、双向、属性索引）