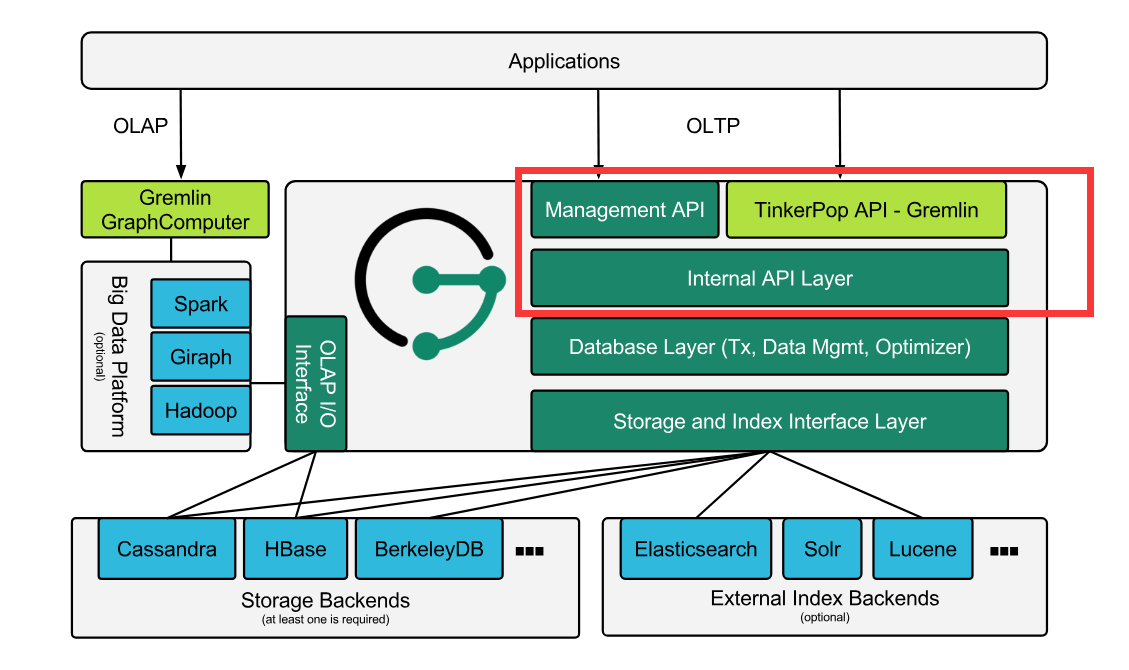
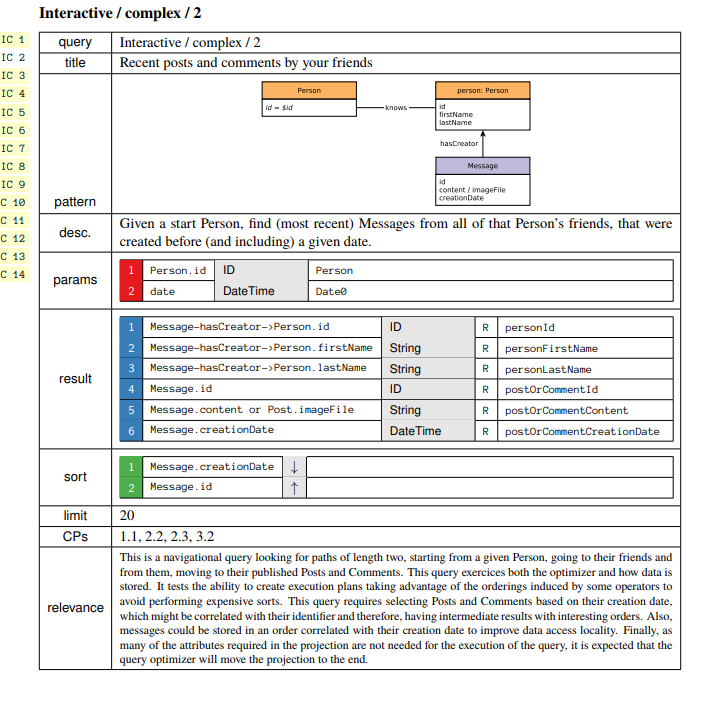
# Janusgraph内部API与Gremlin API

JanusGraph发行版包含一个Gremlin控制台命令行，可以轻松上手并与JanusGraph交互。调用bin/gremlin.sh即可启动控制台，然后使用JanusGraphFactory打开JanusGraph图形。JanusGraphFactory也可用于在基于JVM的用户应用程序中打开嵌入式JanusGraph图形实例。在这种情况下，应用程序可以直接通过其公共API调用JanusGraph。



如上图所示，Gremlin API 位于Internal API Layer上层，直接面向应用程序。Gremlin是Apache TinkerPop的一个组件，它独立于JanusGraph开发，得到大多数数据库的支持。JanusGraph的发行版会自带一个Gremlin。Gremlin是一种面向路径的语言，能够简洁地表达复杂的图形遍历和变异操作。Gremlin是一种功能性语言，便利运算符连接在一起形成类似路径的表达式。ldbc\_snb\_driver中定义了对于interactive workload的基本的查询与更新操作。基于给定的查询操作，我们基于gremlin java api 实现了相关的handler。以其中的中的一个query2为例，如下图所示：



根据描述，此条查询的目的是给定一个起始的people，寻找其朋友在给定的日期前发布的消息，返回的结果为personId，personFirstName，personLastName，postOrCommentId，personOrCommentContent和postOrCommentCreationDate。ldbc\_snb\_driver对每一个查询的返回结果定义了一个接收类，query2的数据结构定义在LdbcQuery2Result中。查询语句如下：

query="g.V().has('Person.id', $id)."+

"out('knows').as('friend').valueMap().as('x').in('hasCreator').has('creationDate',P.lte($maxDate))."+

"order().by('creationDate',decr).by('messageId',incr)."+

"limit($limit).as('post').valueMap().as('y')"+

".select('x','y')\n";

通过

ResultSet resultSet = dbConnectionState.runQuery(query, parameters);

即可获得结果集，使用到的接口在org.apache.tinkerpop.gremlin中，具体实现可参考程序代码。

下图为benchmark给出的测试结果：

