解决跨分片的SQL JOIN的问题，远比想象的复杂，而且往往无法实现高效的处理，既然如此，就依靠人工的智力，去编程解决业务系统中特定几个必须跨分片的SQL的JOIN逻辑，MyCAT提供特定的API供程序员调用，这就是MyCAT创新性的思路——人工智能。

以一个跨节点的SQL为例，

Select a.id,a.name,b.title from a,b where a.id=b.id

 其中a在分片1，2，3上，b在4，5，6上，需要把数据全部拉到本地（MyCAT服务器）,执行JOIN逻辑，具体过程如下（只是一种可能的执行逻辑）：

EngineCtx ctx=new EngineCtx();//**包含** MyCat.SQLEngine

String sql=,“select a.id ,a.name from a ”;

//在a表所在的所有分片上顺序执行下面的本地SQL

ctx. executeNativeSQLSequnceJob(allAnodes,new DirectDBJoinHandler());

DirectDBJoinHandler类是一个回调类，负责处理SQL执行过程中返回的数据包，这里的这个类，主要目的是用a表返回的ID信息，去b表上查询对于的记录，做实时的关联：

DirectDBJoinHandler{

 Private HashMap<byte[],byte[]> rows;//Key为id,value为一行记录的Column原始Byte数组，这里是a.id,a.name,b.title这三个要输出的字段

 Public Boolean onHeader(byte[] header)

{

//保存Header信息，用于从Row中获取Field字段值

}

 Public Boolean onRowData(byte[] rowData)

{

 String id=getColumnAsString(“id”);

//放入结果集, b.title字段未知，所以先空着

rows.put(getColumnRawBytes(“id”), rowData);

 //满1000条，发送一个查询请求

 String sql=”select b.id, b.name from b where id in (………….)”;

//此SQL在B的所有节点上并发执行，返回的结果直接输出到客户端

 ctx. executeNativeSQLParallJob (allBNodes, sql ,new MyRowOutPutDataHandler(rows));

}

 Public Boolean onRowFinished()

 {

 }

Public void onJobFinished()

 {

 If(ctx.allJobFinished())

 {///used total time ….

 }

}

}

/最后，增加一个Job事件监听器，这里是所有Job完成后，往客户端发送RowEnd包，结束整个流程。

ctx.setJobEventListener(new JobEventHandler(){public void onJobFinished(){ client.writeRowEndPackage()}});

以上提供一个SQL执行框架，完全是异步的模式执行，并且以后会提供更多高质量的API，简化分布式数据处理，比如内存结合文件的数据JOIN算法，分组算法，排序算法等等，

期待更多的牛人一起来完善。