# Android ContentProvider共享数据的原理

参考：<https://blog.csdn.net/Luoshengyang/article/details/6967204>

1. Android应用程序之间不能直接访问对方的数据文件的障碍在于每一个应用程序都有自己的用户ID，而每一个应用程序所创建的文件的读写权限都是只赋予给自己所属的用户，因此就限制了应用程序之间相互读写数据的操作;
2. Binder进程间通信机制可以突破了以应用程序为边界的权限控制来实现在不同应用程序之间传输数据，而Content Provider组件在不同应用程序之间共享数据正是基于Binder进程间通信机制来实现的。
3. Content Provider组件在不同应用程序之间传输数据正是基于匿名共享内存机制来实现的。
4. 第三方应用程序通过ContentProviderProxy.query()访问Content Provider中的数据时，首先会在本进程中创建一个CursorWindow对象，CursorWindow内部通过CursorWindow.initBuffer()创建了一块匿名共享内存，同时实现了Parcel接口，因此它可以在进程间传输。然后第三方应用程序把此CursorWindow对象（连同它内部的匿名共享内存文件描述符）通过Binder进程间调用传输到Content Provide中。这个匿名共享内存文件描述符传输到Binder驱动程序时，Binder驱动程序会在目标进程（即Content Provider所在的进程）中创建另一个匿名共享文件描述符，指向前面已经创建好的匿名共享内存，从而实在两个进程中共享同一块匿名内存;
5. 在Content Provider中，利用在Binder驱动程序为其创建好的这个匿名共享内存文件描述符，在本进程中创建了一个CursorWindow对象。
6. ContentProviderProxy.bulkQueryInternal()把查询参数都写到一个Parcel对象中，然后通过Binder进程间通信机制把查询请求传给Content Provider处理；得到一个Binder类型的IBulkCursor接口（其实际指向Content Provider的CursorToBulkCursorAdaptor对象），第三方应用通过这个IBulkCursor接口，就可以利用Binder进程间调用来访问从Content Provider中查询得到的数据了；
7. 在ContentProviderNative.onTransact()中把查询请求参数从data中读取出来，重建一个CursorWindow对象并用data中的数据初始化该CursorWindow，从而使该CursorWindow与第三方应用中创建的CursorWindow的共享内存为同一块共享内存；
8. Transport.bulkQuery()调用ContentProvider.query()构造一个数据库查询计划，即准备好SQL语句，并通过IBulkCursor接口将查询的数据存放到CursorWindow的共享内存中，第三方应用通过这个IBulkCursor接口获取从Content Provider中查询到的数据；
9. Content Provider向第三方应用程序返回的数据实际上是一个SQLiteCursor对象；