MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

F06_a

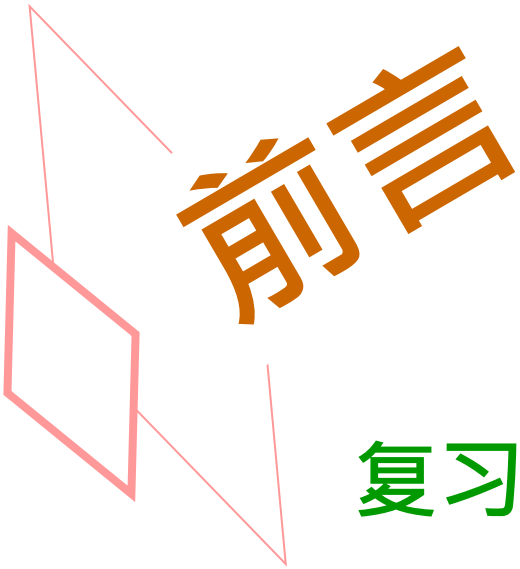
观摩：ContentProvider架构與DB引擎移植方法(a)

By 高煥堂

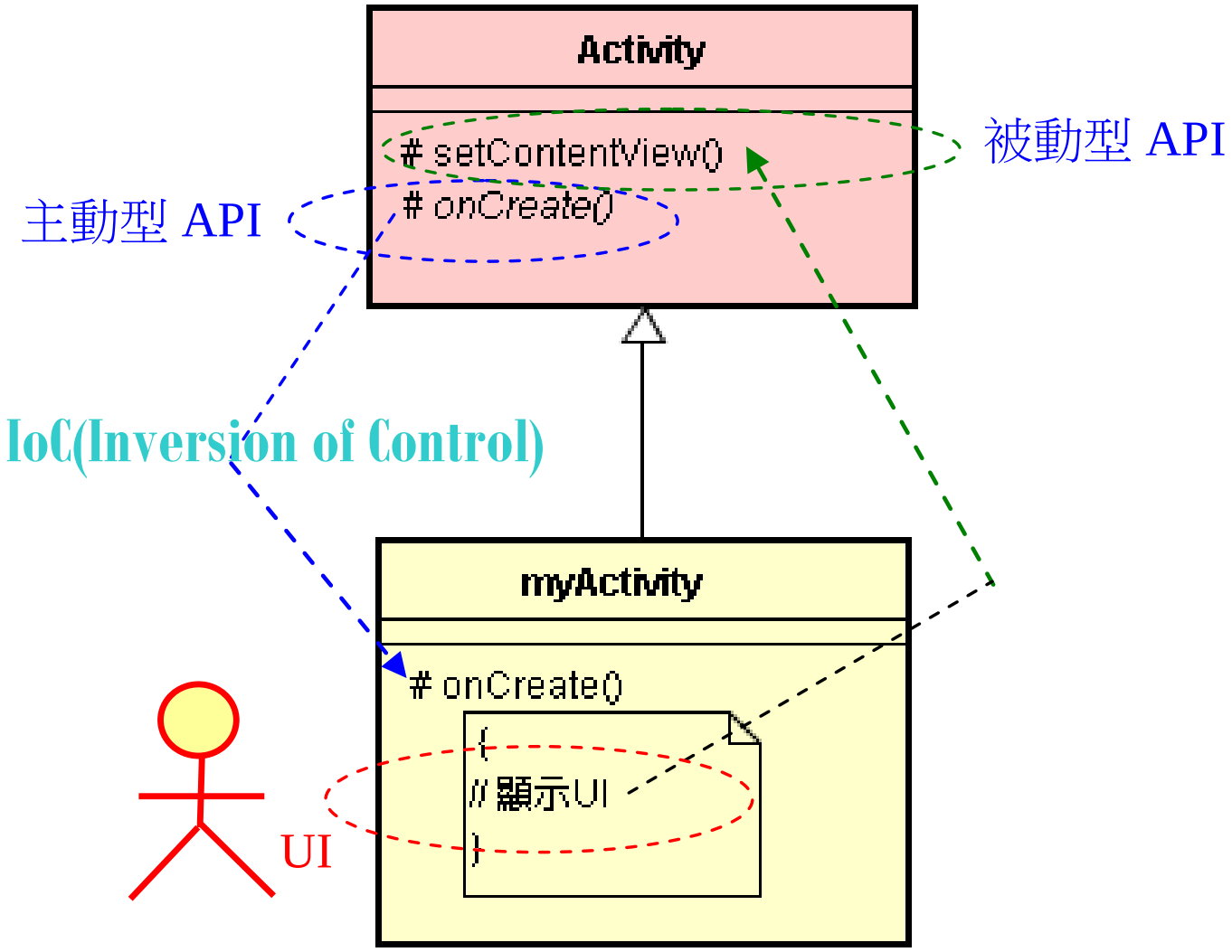
内容

1. 如何保护DB引擎的变动自由度?
2. 从Cursor接口谈起
3. 通用性接口Cursor的使用范例
4. 通用性基类ContentProvider的使用范例
5. 展现DB引擎的变换自由度：以Linter引擎的移植为例

1、如何保护DB引擎的变动自由度?



复习：主动型 vs. 被动型API


Activity
setContentView()
onCreate()
被動型 API
主動型 API
IoC(Inversion of Control)
myActivity
# onCreate()
{
// 顯示UI
}
UI

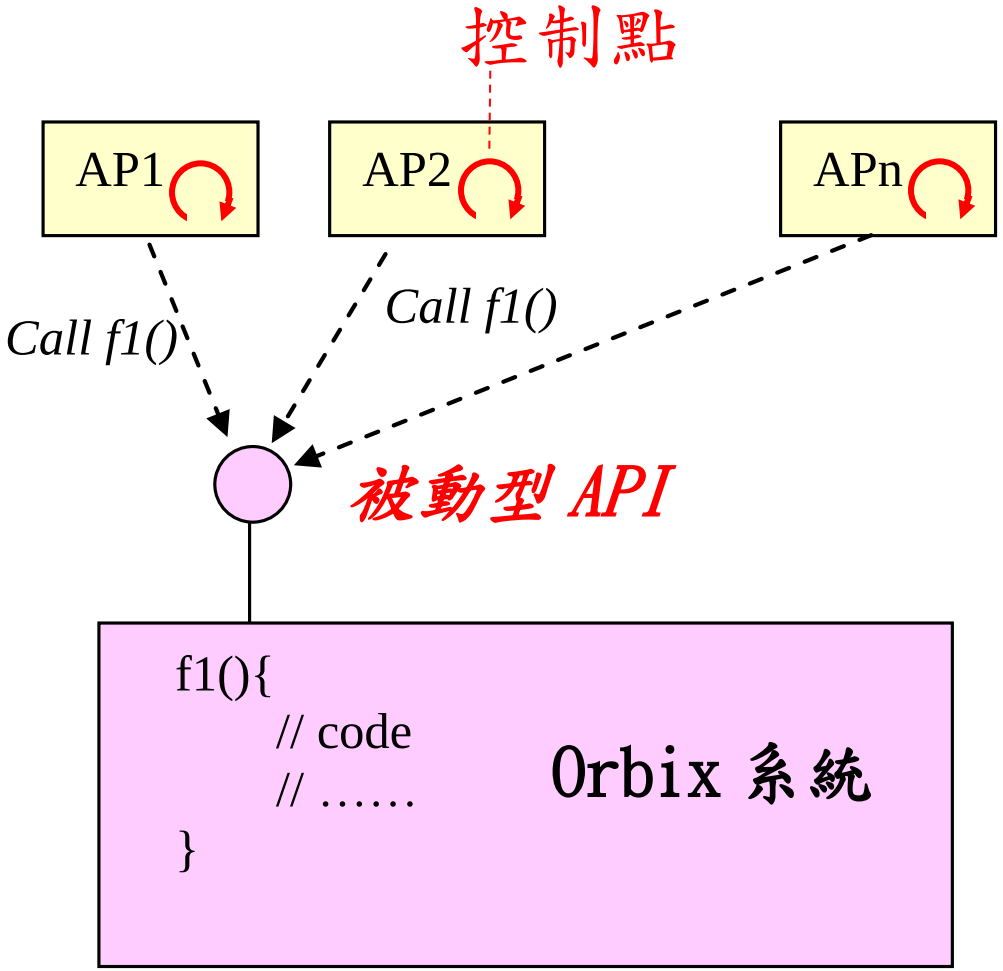
API的分类

- API这个名词，有3个密切关联的动词：
  定义(Define)
  实作(Implement)
  呼叫(Invoke or Call)

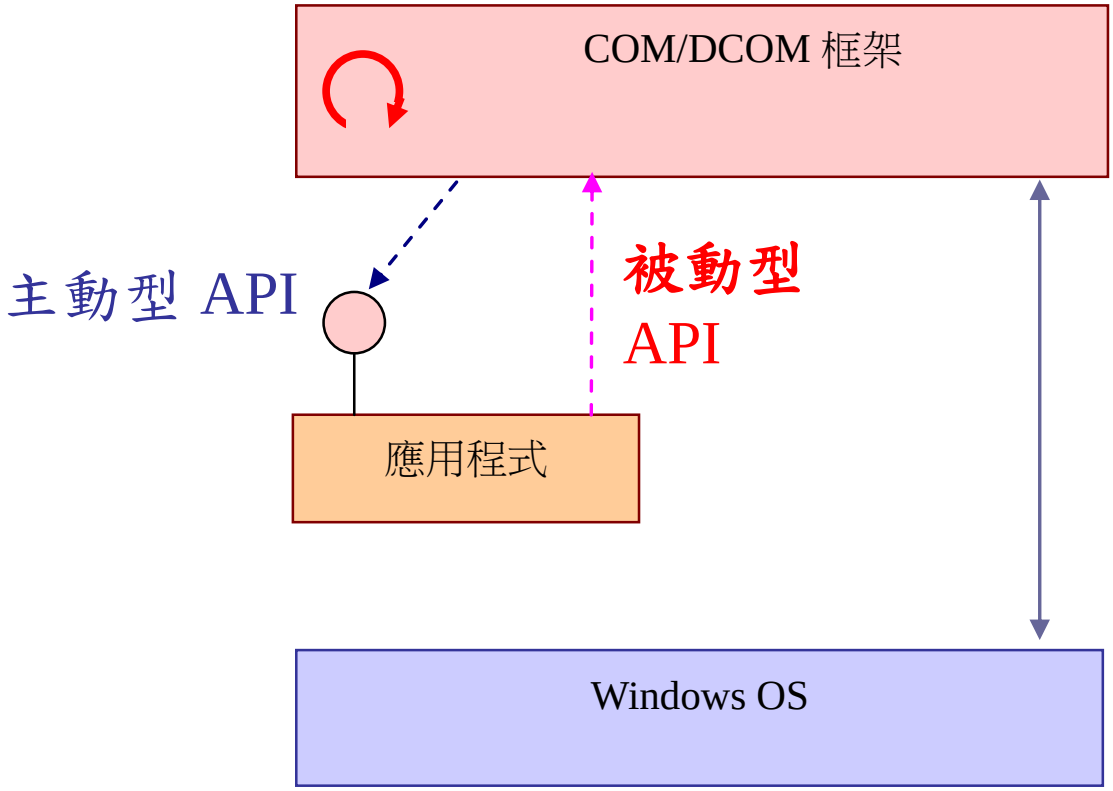
- 根据这3个角度，可将API区分为「主动型」与「被动型」两种。

API类型	
被动型API	•我(即强龙)定义API •且我来实作API •而让对方(即地头蛇)来呼叫我
主动型API	•我定义API •由地头蛇来遵循、实作API •让我来呼叫对方

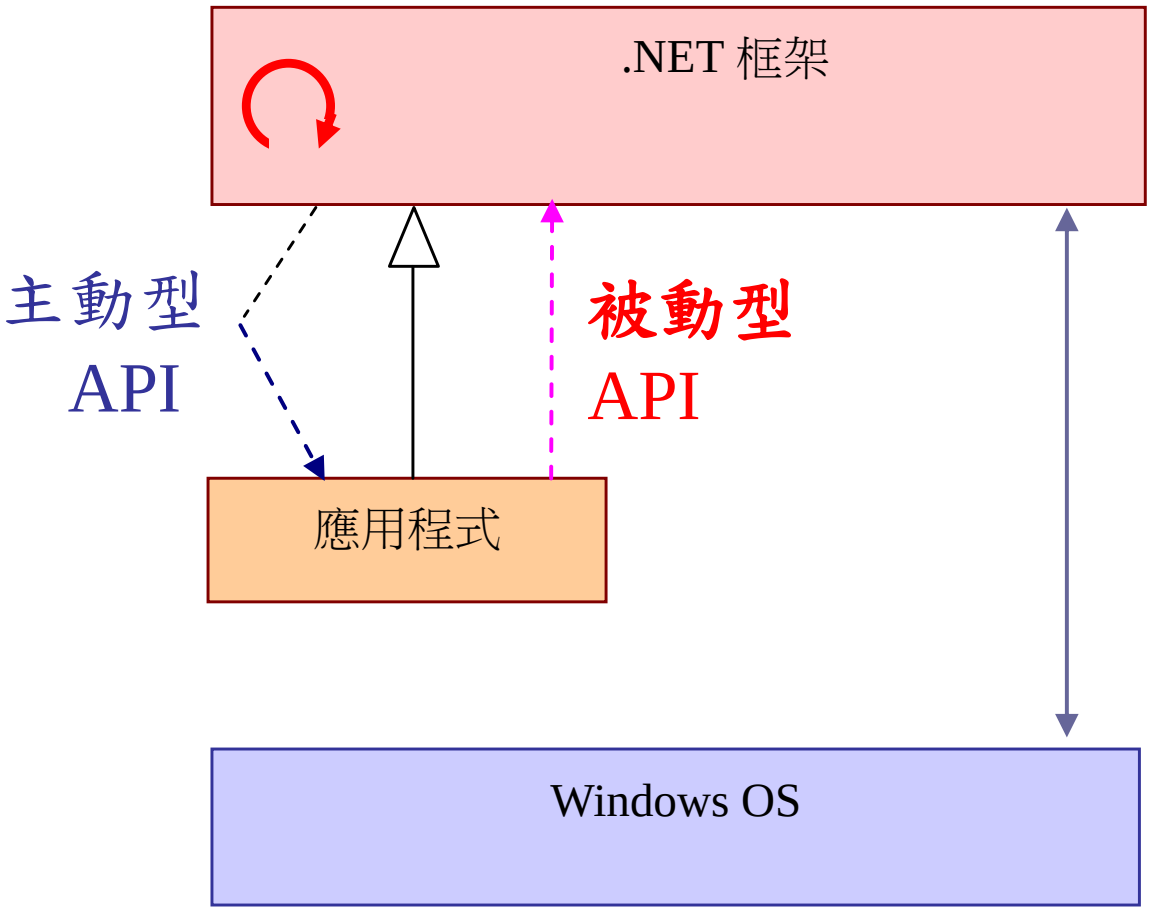
回顧：1990年代初的CORBA和Orbix



1995年的COM/DCOM

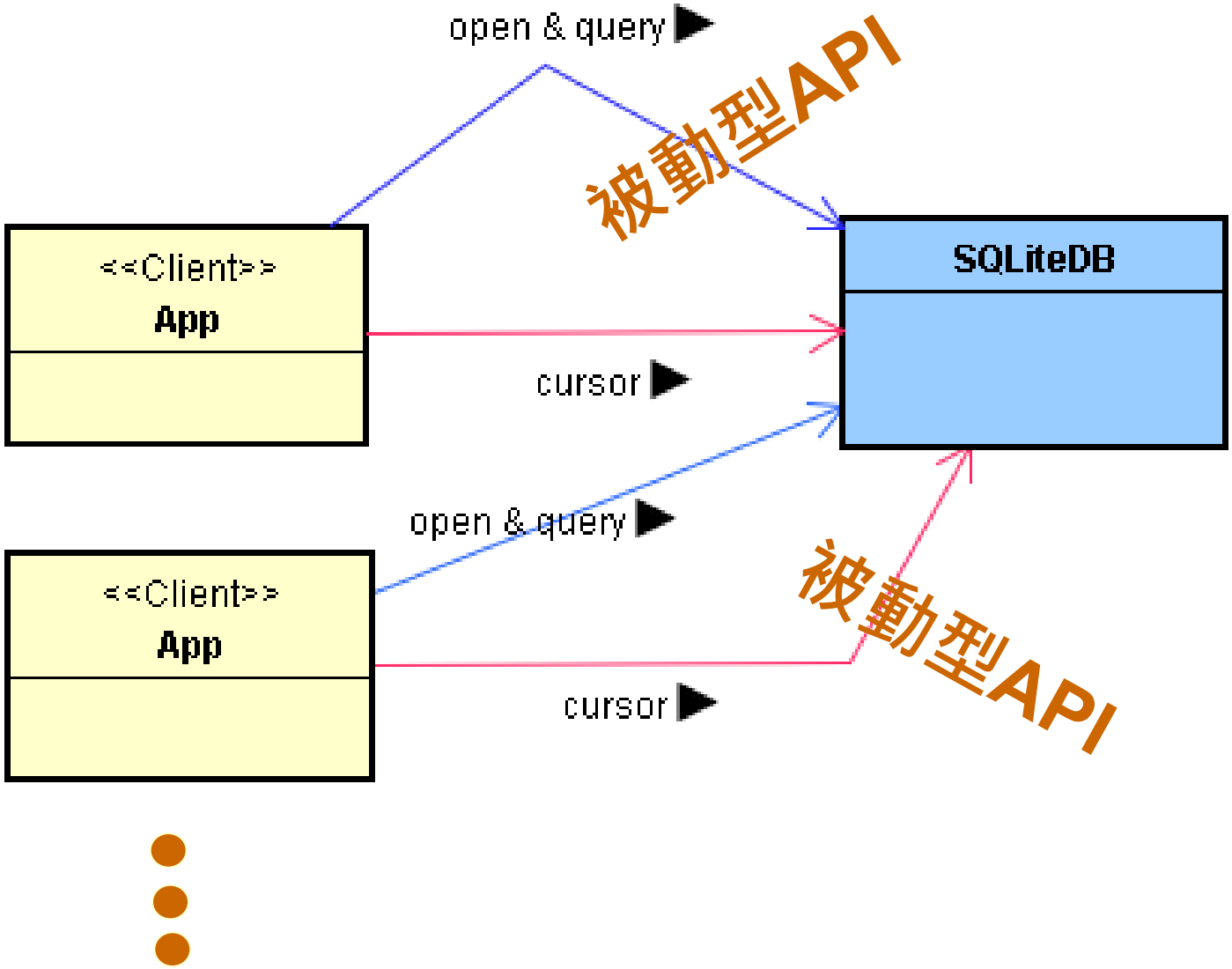


2001年的.NET

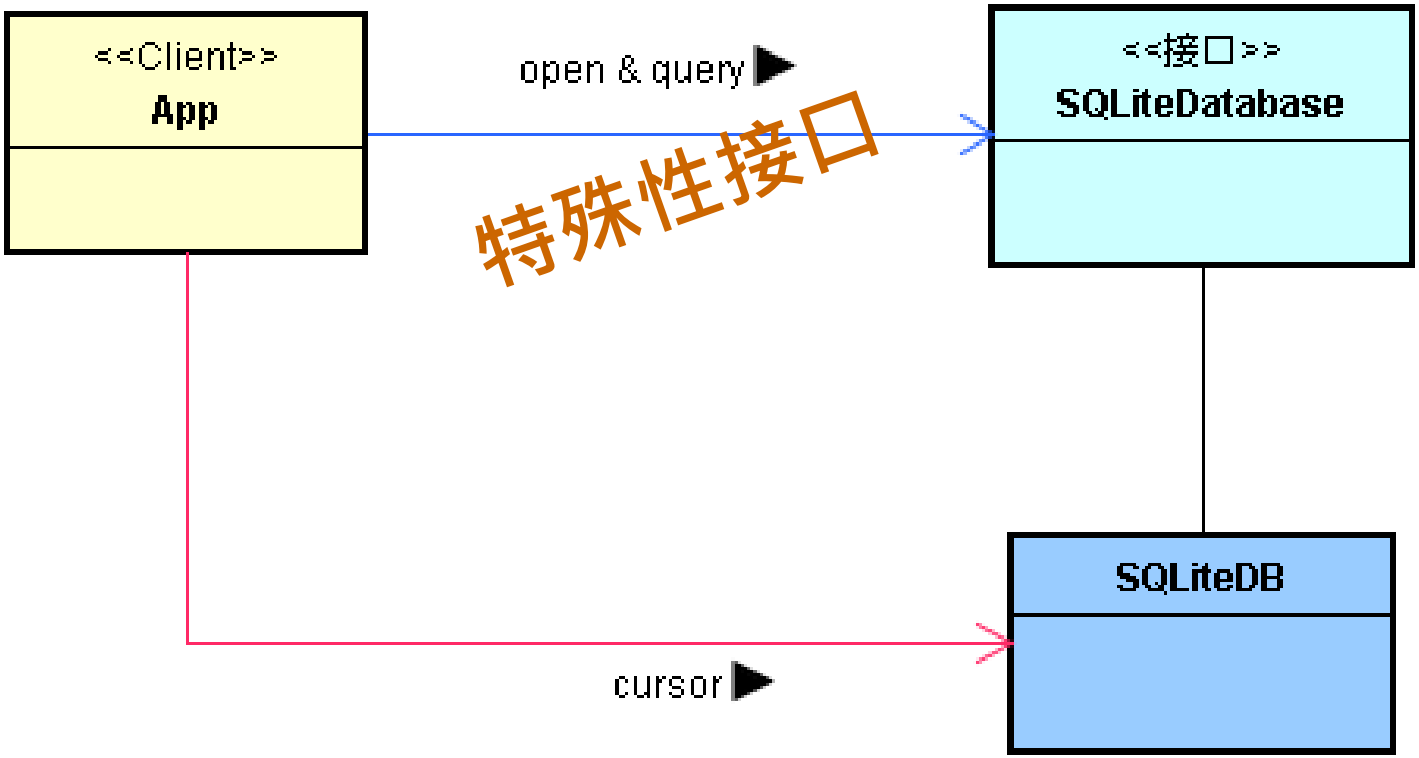


保护DB引擎的变动自由度

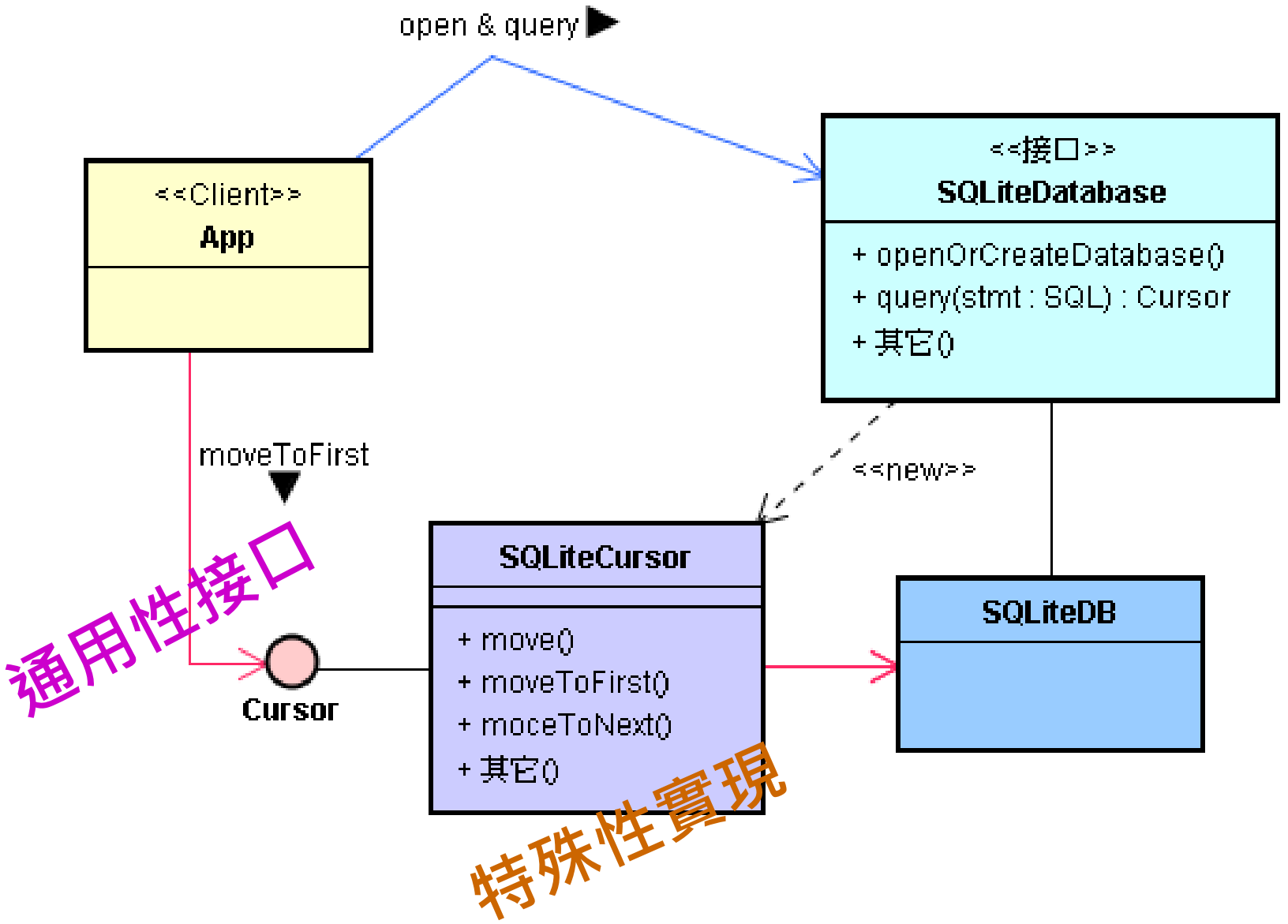
- 毫无保户的情形：让各Client好无限制地使用DB引擎的接口。
- 这种接口，就DB引擎而言，都属于被动型API，受制各Client端，严重伤害DB引擎的变动自由度，局限了DB引擎的成长空间。


open & query
被動型API
<<Client>>
App
SQLiteDB
cursor
open & query
<<Client>>
App
被動型API
cursor

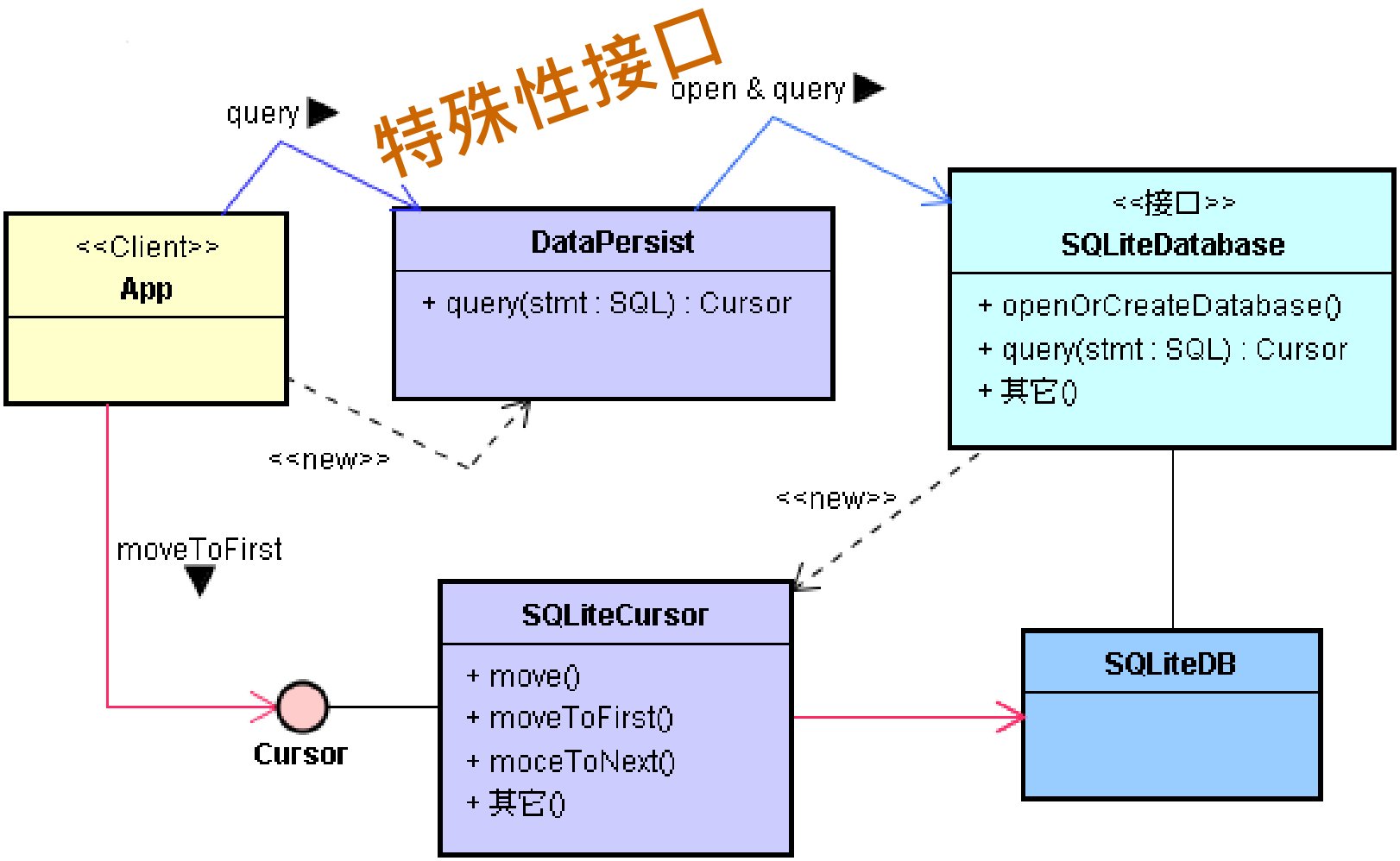
- 最常见的对策就是：DB引擎开发者自己定义一个<接口类>，提供一个对外的接口，隐藏了DB引擎本身的接口。


<<Client>>
App
open & query
<<接口>>
SQLiteDatabase
特殊性接口
SQLiteDB
cursor

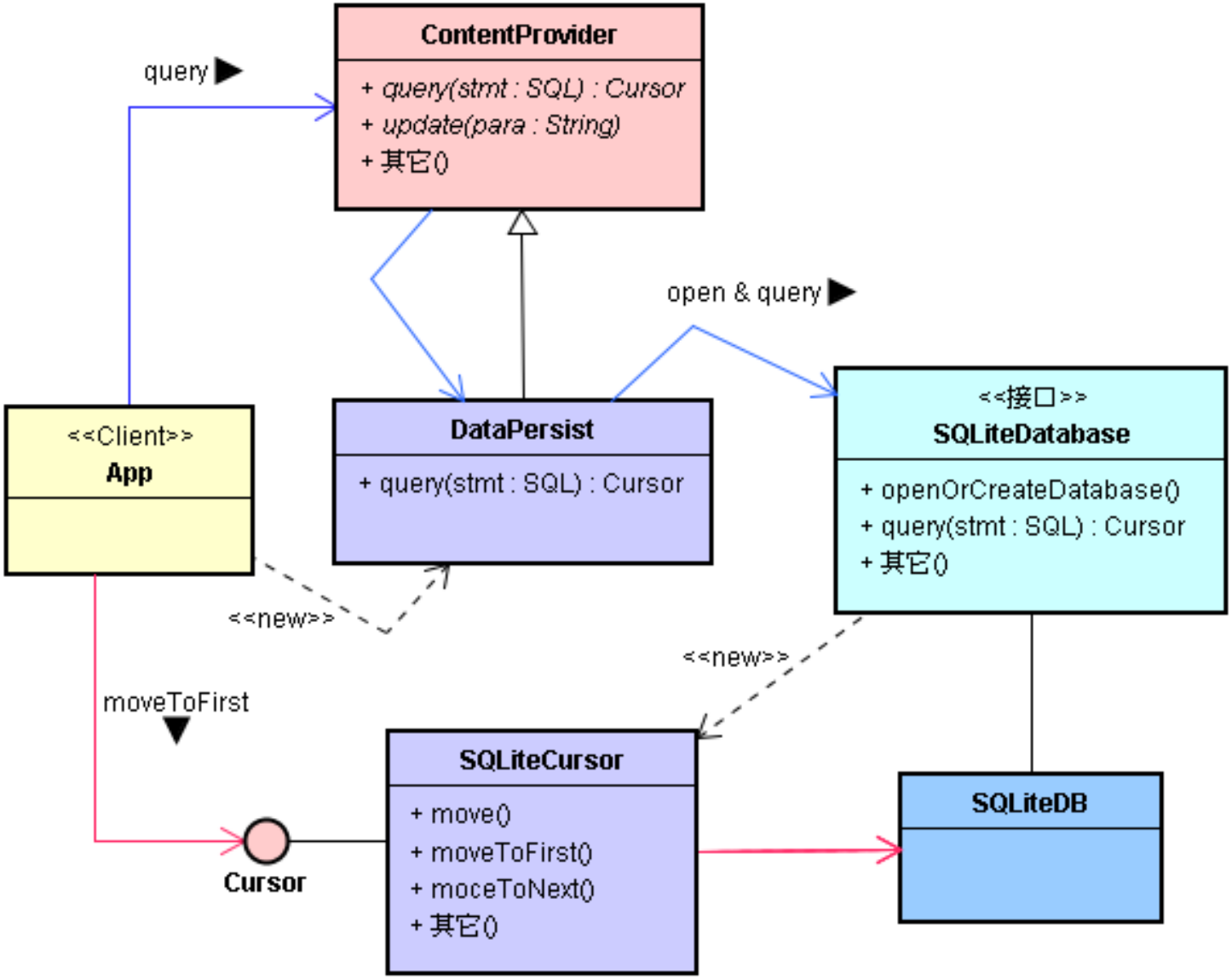
- 然而，这个<接口类>属于特殊性接口，不同DB引擎的厂商，都有专属的<接口类>；所以不是各DB引擎都适合的通用性接口。
- 比較美好的絕對策是：提供通用性接口給Client端。

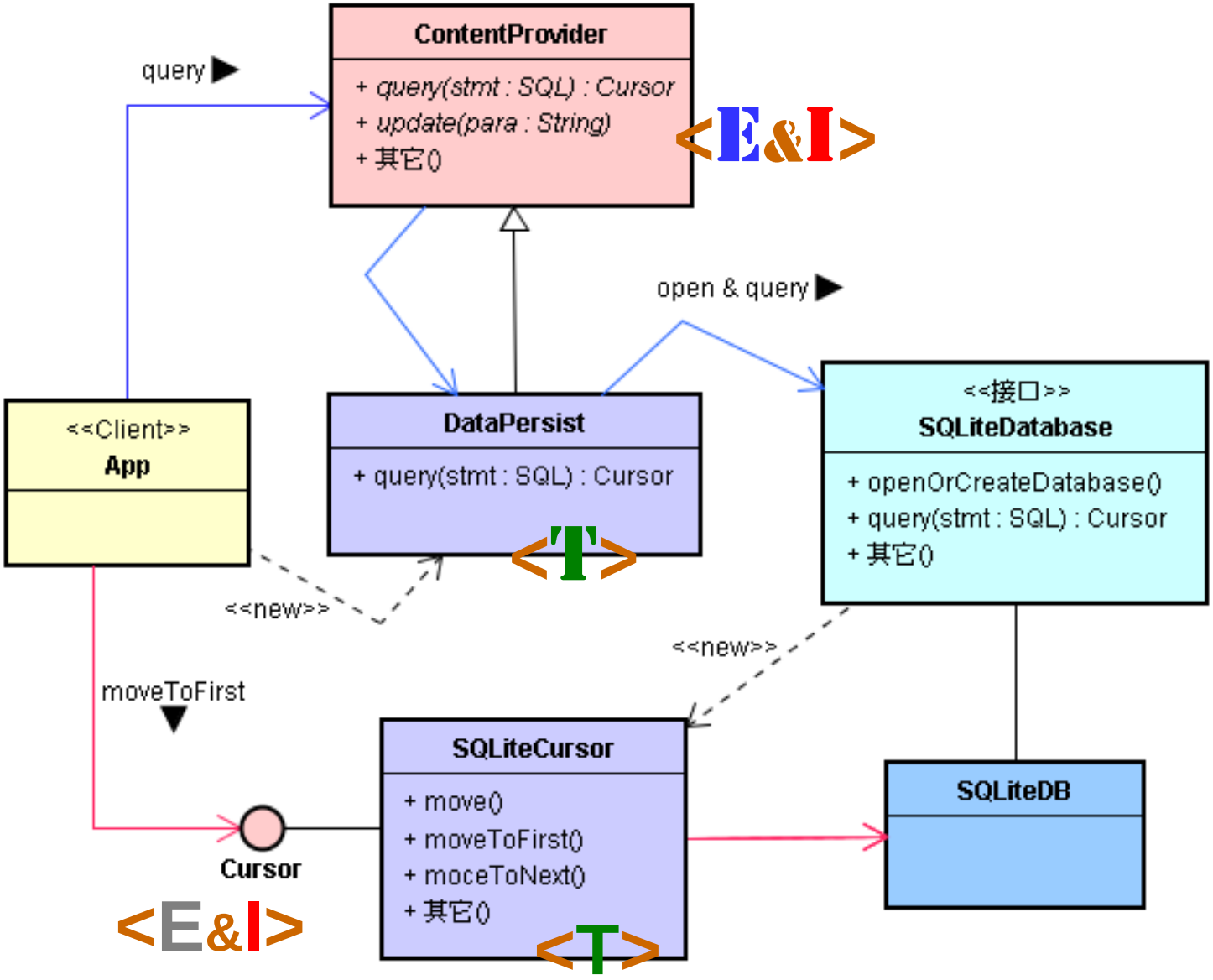

open & query
<<Client>>
App
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
moveToFirst
<<new>>
通用性接口
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor
特殊性實現

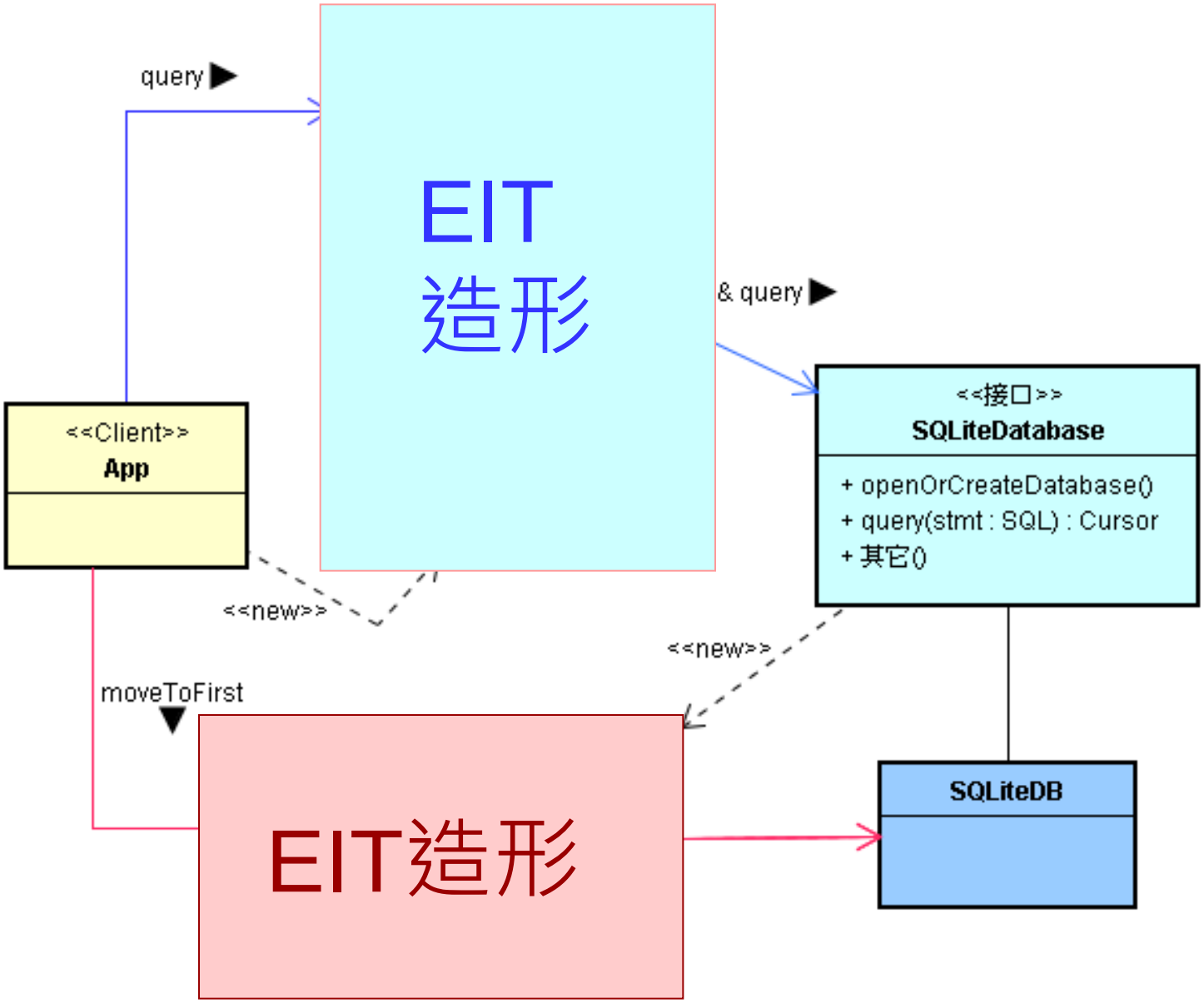
- 上图解决了一半的设计问题，对于”open & query”的交互，还是透过特殊性接口。即使再增添一个DataPersist类(如下图)，仍然是一样的问题。


特殊性接口
query
open & query
<<Client>>
App
DataPersist
+ query(stmt : SQL) : Cursor
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
<<new>>
moveToFirst
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor

- 解决之道是：让特殊性的DataPersist类，来实践一个通用性的<E&I>。Android提供一个ContentProvider基类，就是这个通用性<E&I>角色。


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
query
open & query
<<Client>>
App
DataPersist
+ query(stmt : SQL) : Cursor
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
<<new>>
moveToFirst
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
<E&I>
query
open & query
<<Client>>
App
DataPersist
+ query(stmt : SQL) : Cursor
<T>
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
<<new>>
moveToFirst
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
<T>
SQLiteDB
Cursor
<E&I>


query
EIT
造形
& query
<<接口>>
SQLiteDatabase
+ openOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<Client>>
App
<<new>>
<<new>>
moveToFirst
EIT造形
SQLiteDB

~ Continued ~