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

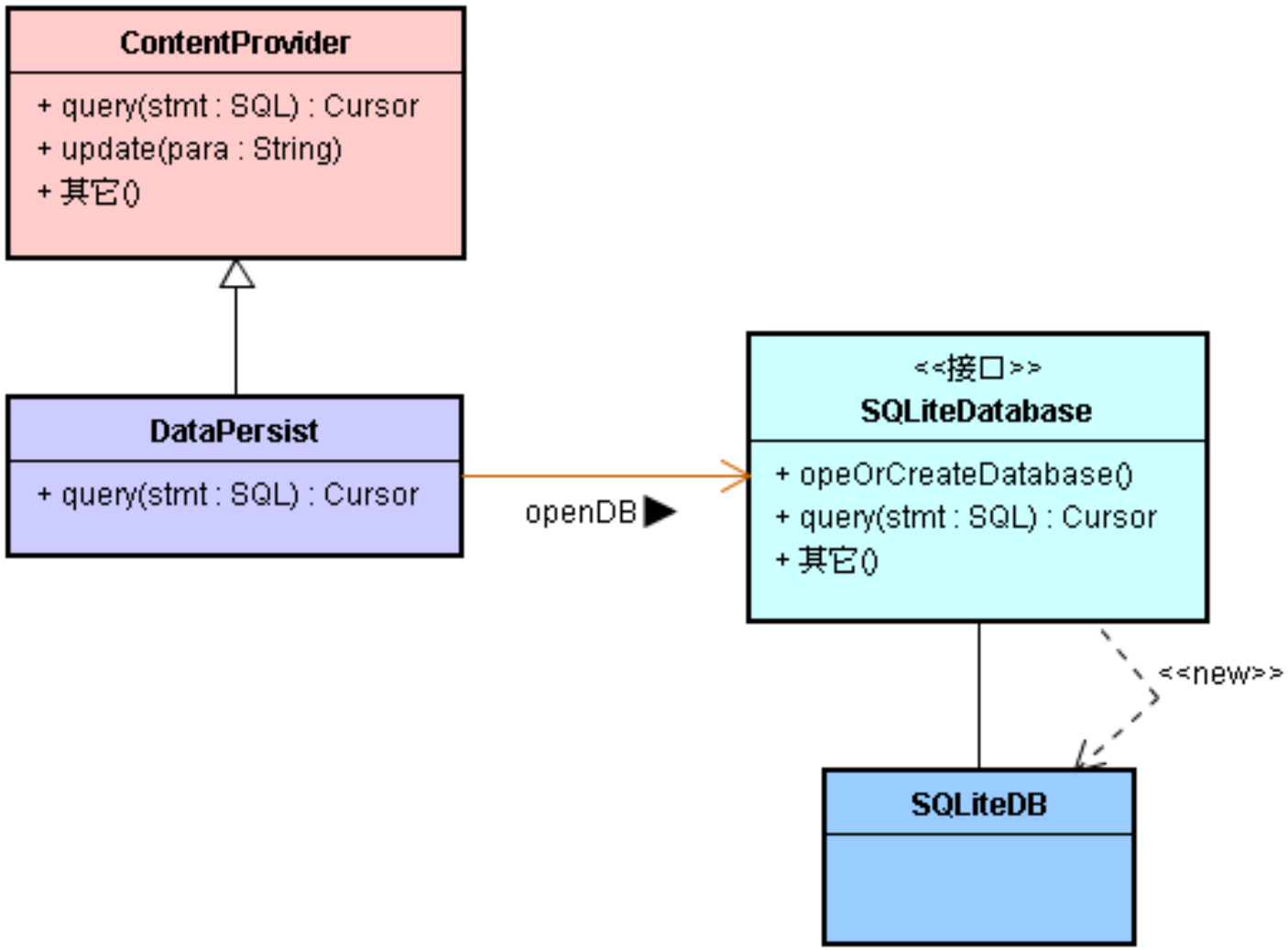
F06_c

观摩：ContentProvider架构與DB引擎移植方法(c)

By 高煥堂

4、通用性基类ContentProvider的使用范例

- 刚才的范例里，我们直接使用DataPersist类的接口来与SQLite沟通。
- 本节将替DataPersist配上ContentProvider基类，让Client能透过ContentProvider新接口来沟通。

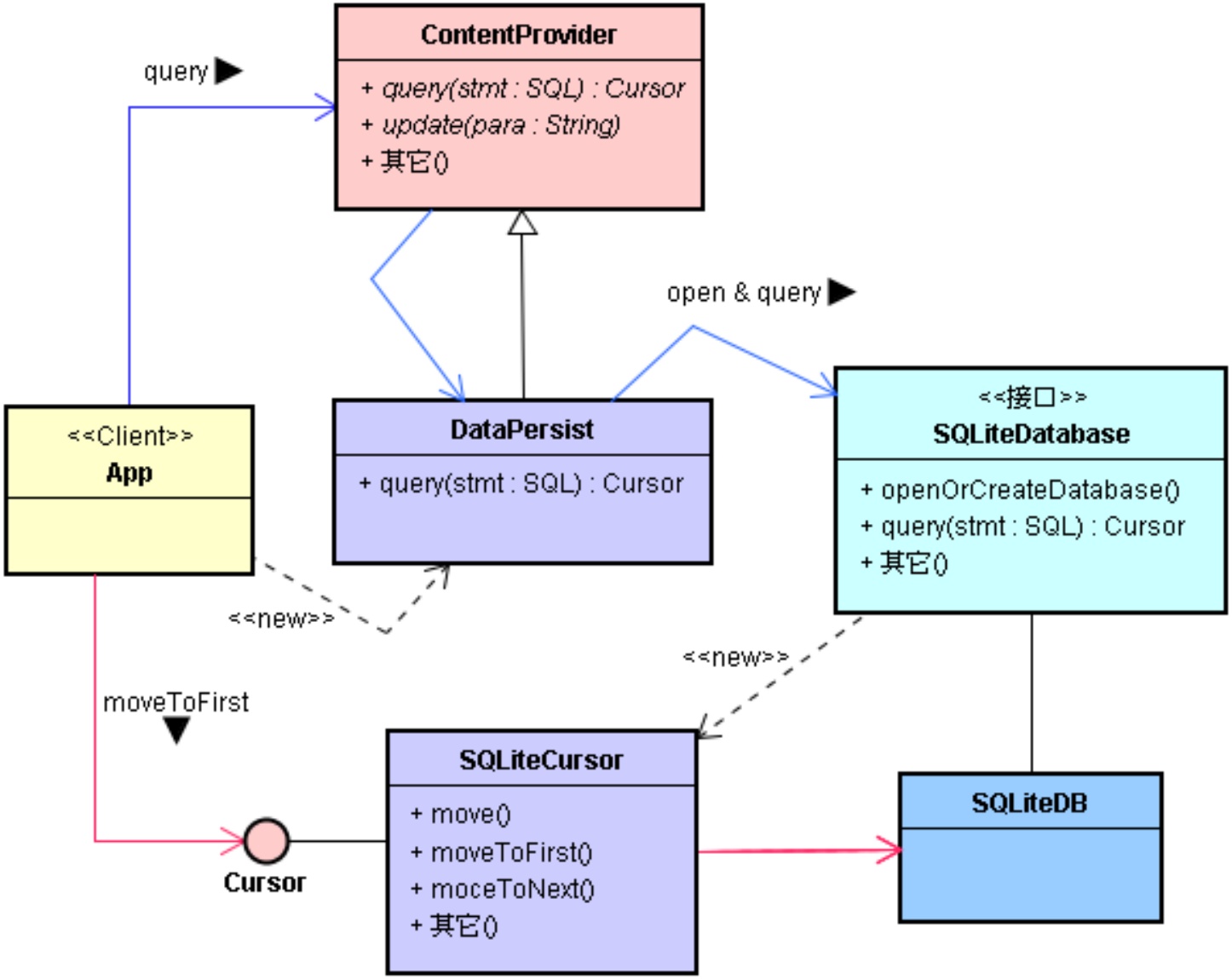

ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
DataPersist
+ query(stmt : SQL) : Cursor
openDB
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
SQLiteDB

ContentProvider义了多个函数，包括：

- query()函数-- 它查询出合乎某条件的数据。
- insert()函数-- 它将存入一笔新资料。
- delete()函数-- 它删除合乎某条件的资料。
- update()函数-- 更新某些笔数据的内容。

- DataPersist类实现query()接口，实际呼叫SQLite数据库的功能。
- 也就是说，Client程序透过ContentProvider接口间接呼叫到DataPersist的query()函数，然后此query()函数才去查询SQLite的DB内容。

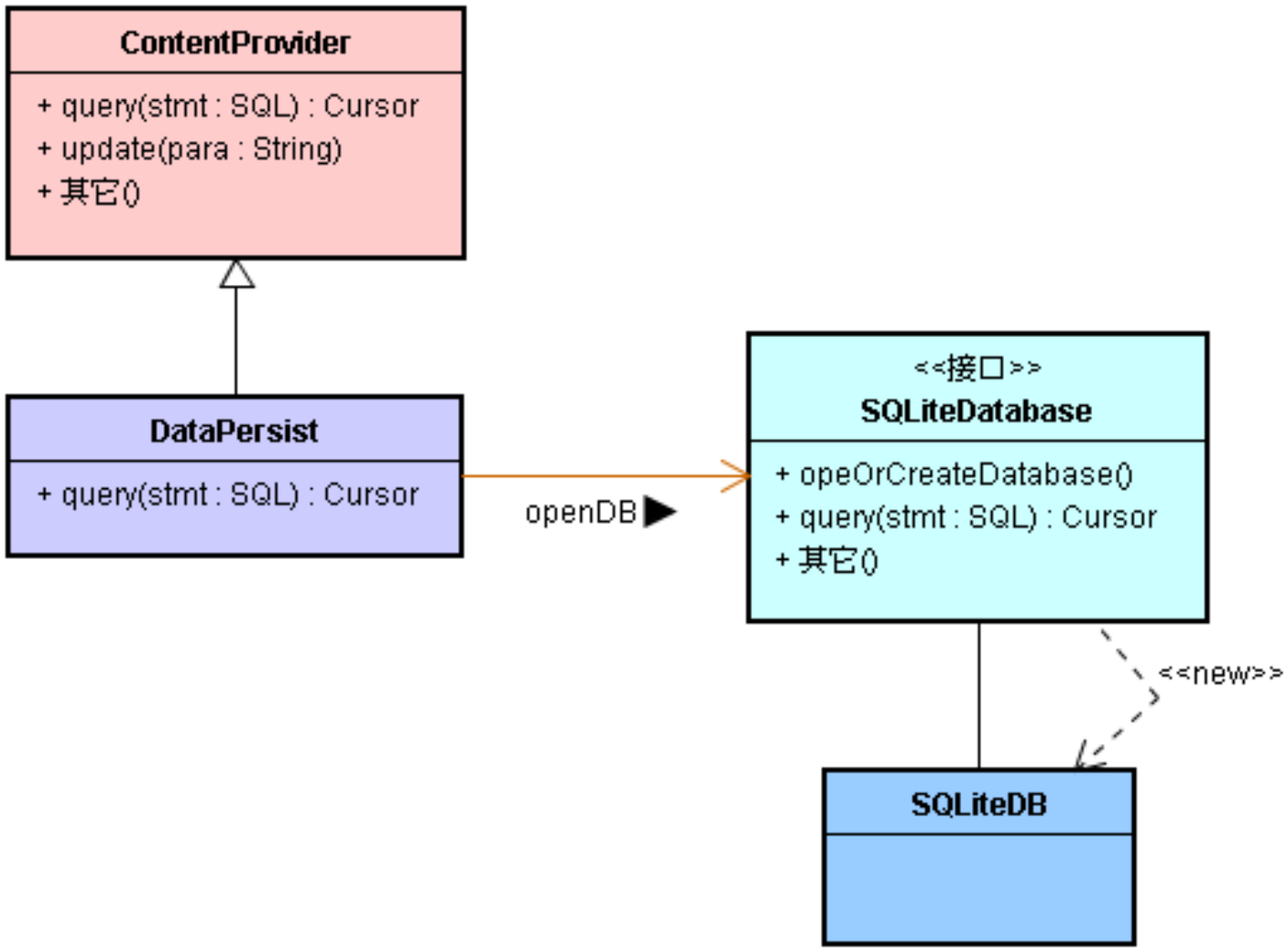
- 查询出来，就诞生一个SQLiteCursor对象，并回传Cursor接口 。
- 让Client程序可藉由Cursor接口来浏览所查询出来的各笔数据。


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

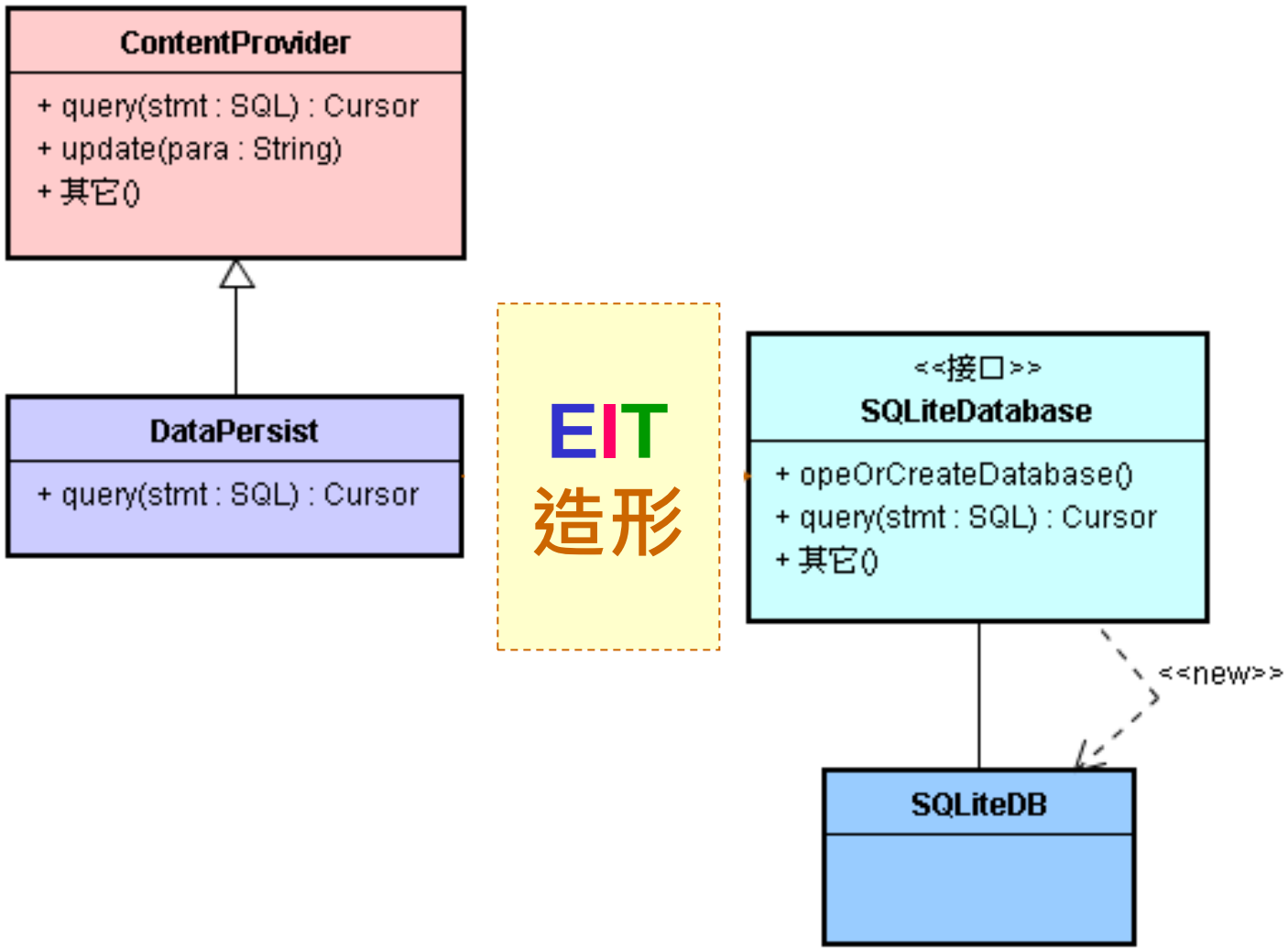
进一步优化架构设计

从DB引擎提供商的视角看

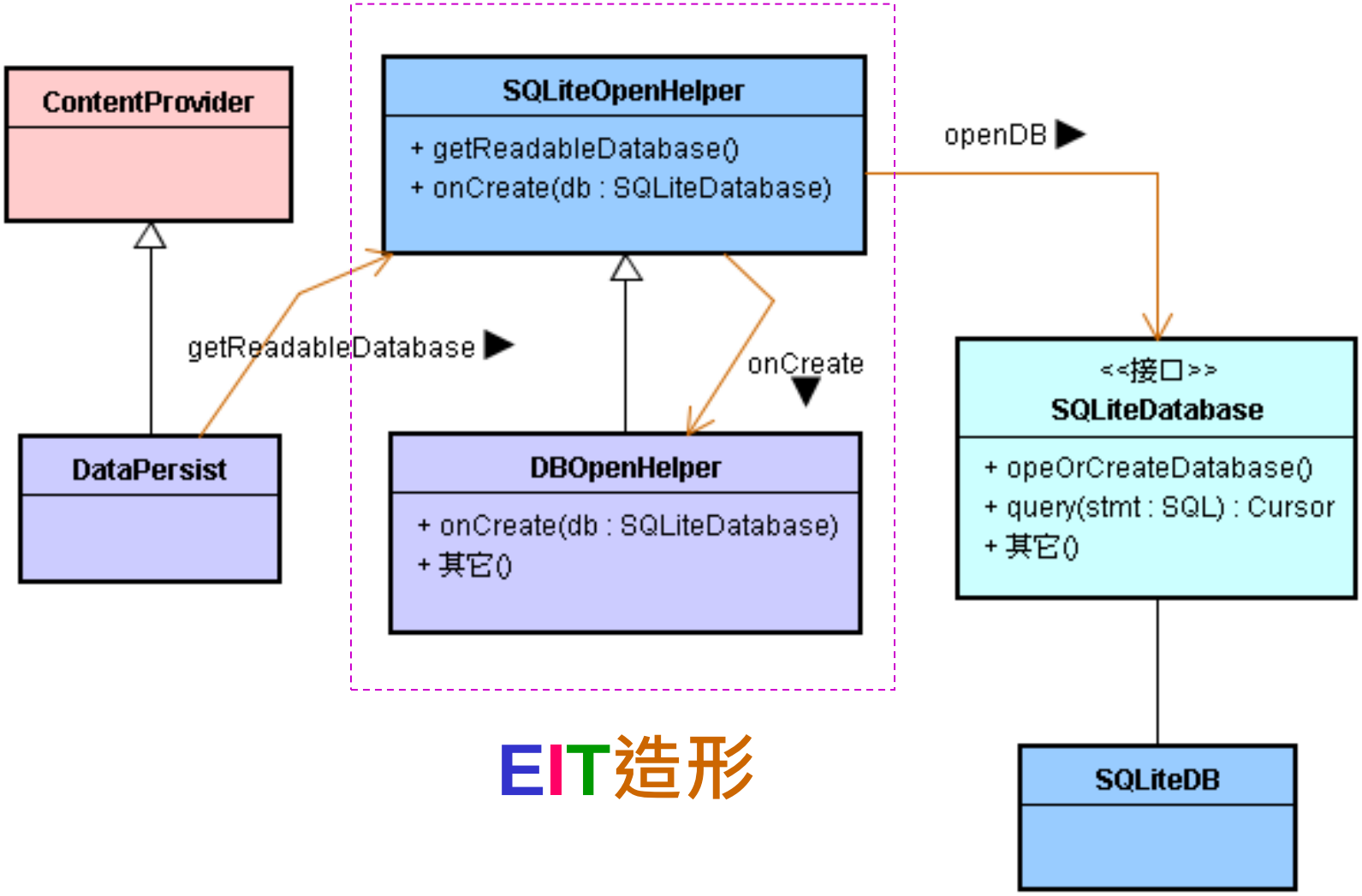
- 在上述的范例中，是由DataPersist类去开启和调用DB引擎的。


ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
DataPersist
+ query(stmt : SQL) : Cursor
openDB
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
SQLiteDB

- 仅仅提供DB引擎的接口，还是不够的。因为这个SQLiteDatabase接口还是被DataPersist所调用的，尤其是DB引擎的开起任务。因而，DB引擎厂商提供接口，只是属于被动型API而已。
- 于是，可擅用EIT造形来提供主动型API。

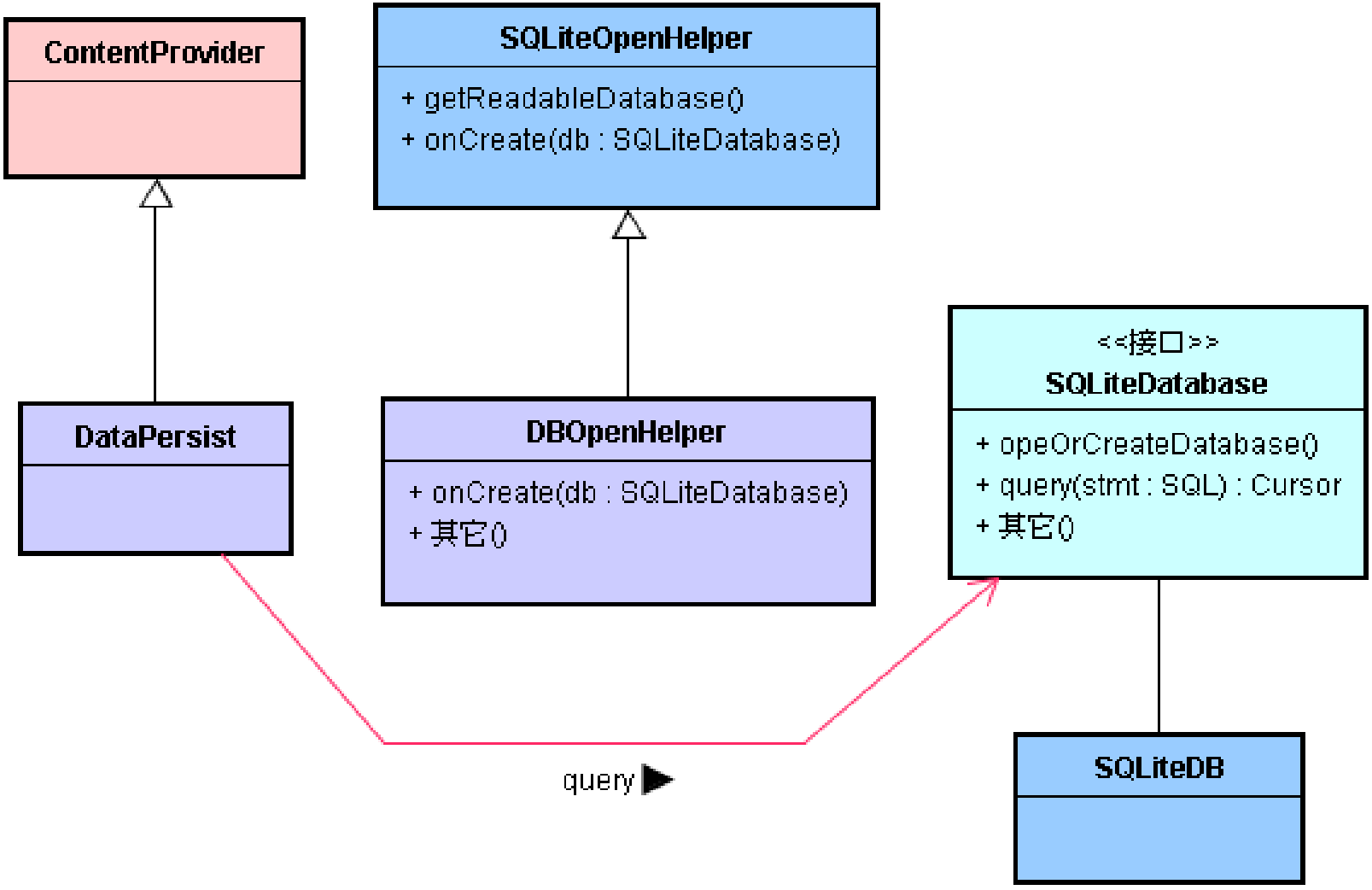

ContentProvider
+ query(stmt : SQL) : Cursor
+ update(para : String)
+ 其它()
DataPersist
+ query(stmt : SQL) : Cursor
EIT
造形
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
SQLiteDB

- 在Android里，这EIT造形的实现类别如下图所示。
- SQLiteOpenHelper类就是<E>角色；而onCreate()就是<I>角色，提供了强势的主动型API。这对DB引擎厂商是有利的，因为SQLiteOpenHelper类有效保护了DB引擎的变动自由度；让DB引擎厂商能够实现”没钱就改版、改版就有钱”的商业策略。

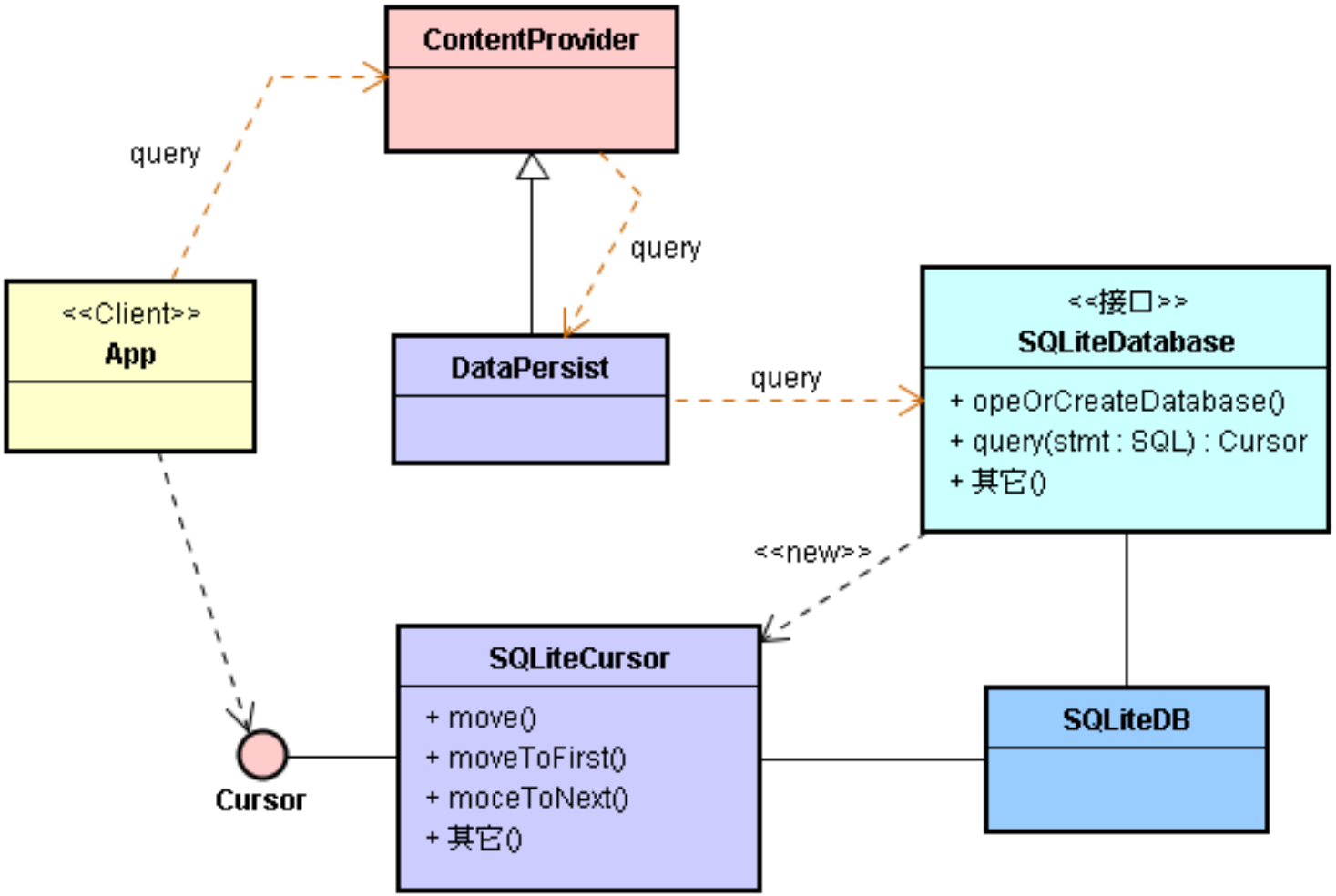

ContentProvider
SQLiteOpenHelper
+ getReadableDatabase()
+ onCreate(db : SQLiteDatabase)
openDB
getReadableDatabase
onCreate
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
DataPersist
DBOpenHelper
+ onCreate(db : SQLiteDatabase)
+ 其它()
SQLiteDB

EIT造形

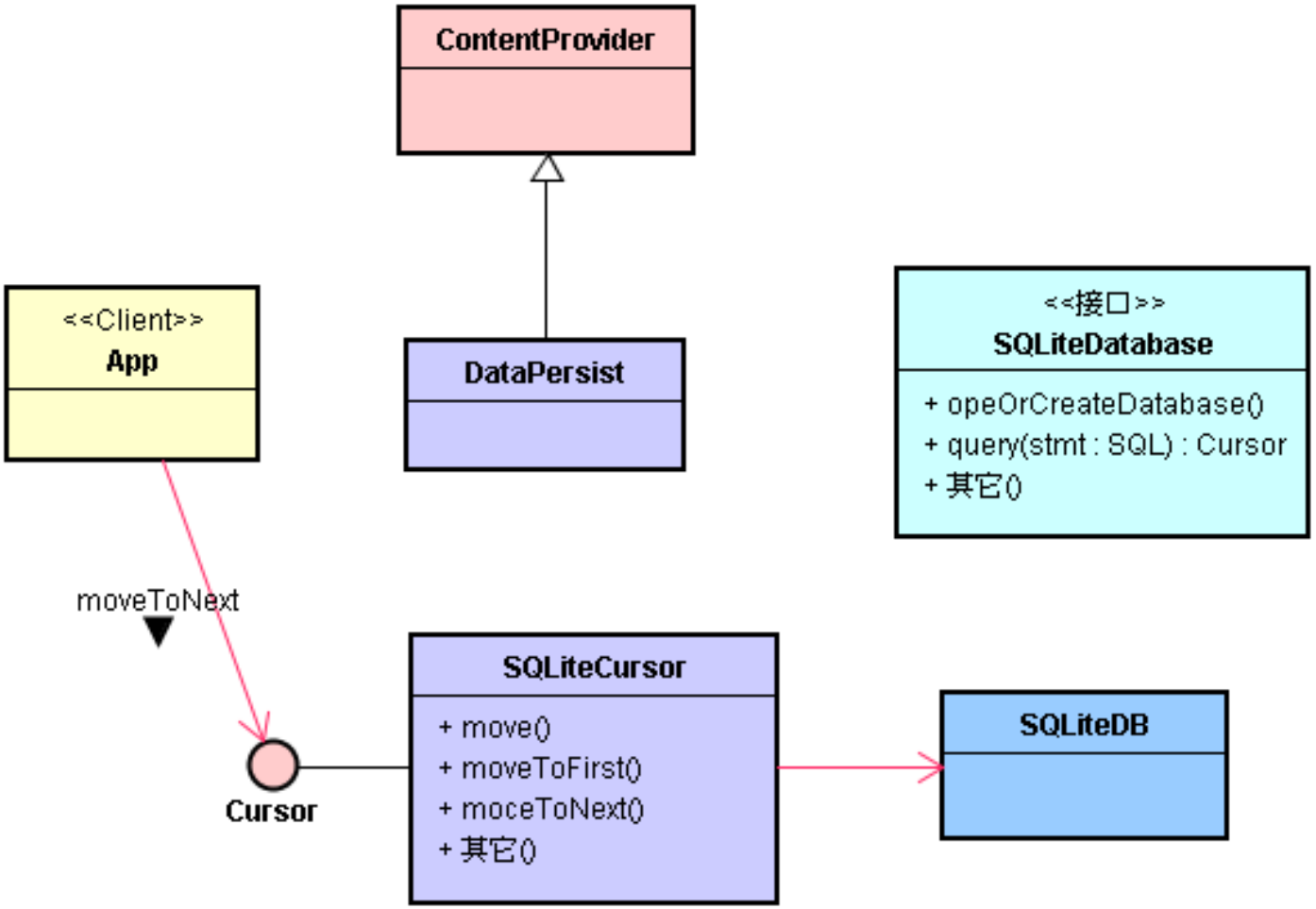
- EIT造形只是在于实现主动型API和创造底层(如DB引擎)变动的自由度而已；它并没有改变DB数据的流动路径，所以没有降低数据的存取效率。


ContentProvider
SQLiteOpenHelper
+ getReadableDatabase()
+ onCreate(db : SQLiteDatabase)
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
DataPersist
DBOpenHelper
+ onCreate(db : SQLiteDatabase)
+ 其它()
query
SQLiteDB

- SQLiteDatabase接口类开启DB，并提供query()函数；在执行query()时就创建一个SQLiteCursor对象，将其Cursor接口回传给Client端。


ContentProvider
query
query
<<Client>>
App
DataPersist
query
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
<<new>>
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor

- Client程序藉由Cursor接口来浏览所查询出来的各笔数据。


ContentProvider
<<Client>>
App
DataPersist
<<接口>>
SQLiteDatabase
+ opeOrCreateDatabase()
+ query(stmt : SQL) : Cursor
+ 其它()
moveToNext
SQLiteCursor
+ move()
+ moveToFirst()
+ moceToNext()
+ 其它()
SQLiteDB
Cursor

- 这也就是目前Android在ContentProvider和DB引擎的整合架构设计了。

范例代码

撰写DataPersist类代码

```
// DataPersist.java
// .....
public class DataPersist extends ContentProvider {
    private static final String DATABASE_NAME = "StudNewDB";
    private static final int DATABASE_VERSION = 2;
    private static final String TABLE_NAME = "StudTable";

    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
            super( context, DATABASE_NAME, null,
                    DATABASE_VERSION);  }
```

```
@Override public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " +TABLE_NAME+" (" +"stud_no"
                    + " TEXT," + "stud_name" + " TEXT" + ");");
    String sql_1 = "insert into " + TABLE_NAME
                    + " (stud_no, stud_name) values('S1001', 'Pam');";
    String sql_2 = "insert into " + TABLE_NAME
                    + " (stud_no, stud_name) values('S1002', 'Steve');";
    String sql_3 = "insert into " + TABLE_NAME
                    + " (stud_no, stud_name) values('S1003', 'John');";
    try { db.execSQL(sql_1);  db.execSQL(sql_2);  db.execSQL(sql_3);
    } catch (SQLException e) { Log.e("ERROR", e.toString());  }
  }
 @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion,
                                                  int newVersion) {}
}
```

```
    private DatabaseHelper mOpenHelper;
   @Override public boolean onCreate() {
          mOpenHelper = new DatabaseHelper(getContext());
          return true;  }
    @Override public Cursor query(Uri uri, String[] projection,
                String selection, String[] selectionArgs, String sortOrder) {
          SQLiteDatabase db = mOpenHelper.getReadableDatabase();
          Cursor c = db.query(TABLE_NAME, projection, null, null,
                                null, null, null);
          return c;   }
    @Override public String getType(Uri uri) {  return null;  }
    @Override public Uri insert(Uri uri, ContentValues initialValues) {
          return uri; }
    @Override public int delete(Uri uri, String where,
                                                String[] whereArgs) {
          return 0; }
    @Override public int update(Uri uri, ContentValues values,
                  String where, String[] whereArgs)   {  return 0;  }
}
```

撰写Activity类代码

```
//  ac01.java
// .......
public class ac01 extends ListActivity {
    public static int g_variable;
    public static final String AUTHORITY = "com.misoo.provider.rx09-02";
    public static final Uri CONTENT_URI =
                Uri.parse("content://" + AUTHORITY + "/Student");
    private static final String[] PROJECTION
              = new String[]{ "stud_no", "stud_name"};
    @Override protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            Intent intent = getIntent();
            if (intent.getData() == null)  intent.setData(CONTENT_URI);
```

```
        Cursor cur = getContentResolver().query(getIntent().getData(),
                        PROJECTION, null, null, null);
        ArrayList<Map<String, Object>> coll
                        = new ArrayList<Map<String, Object>>();
        Map<String, Object> item;
        cur.moveToFirst();
        while (!cur.isAfterLast()) {
            item = new HashMap<String, Object>();
            item.put("c1", cur.getString(0) + ",  " + cur.getString(1));
            coll.add(item);
            cur.moveToNext();
            this.setListAdapter(new SimpleAdapter(this, coll,
                    android.R.layout.simple_list_item_1,
                    new String[] { "c1" }, new int[] { android.R.id.text1 }));
        }
        @Override
        protected void onListItemClick(ListView l, View v, int position,
                                long id) { finish();}
}
```

~ Continued ~