# 1 软件名称

完成人：钟昌宏

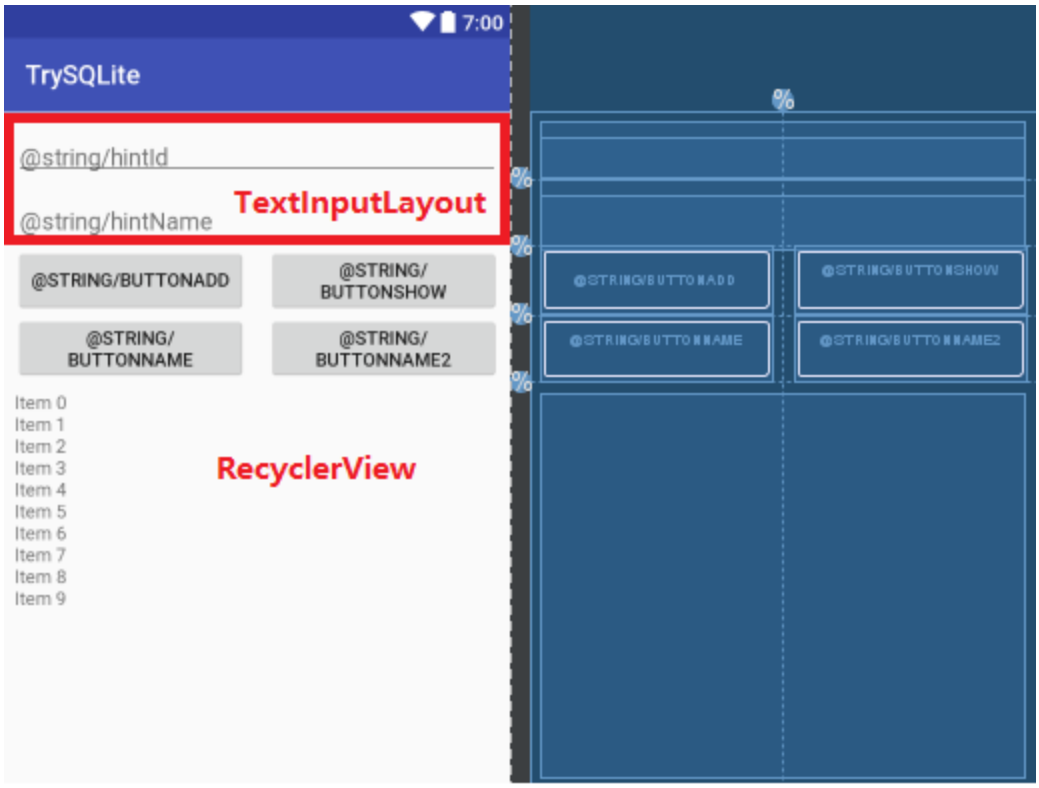
学号：20152100127

完成时间：2017/12/2

# 2 软件内容简介

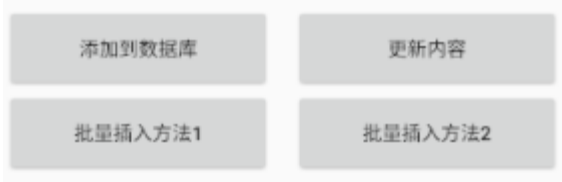
实现数据库的新建，向数据库中插入数据，从数据库读取数据，批量插入数据等功能，还进行了两种批量插入方法的耗时比较。

# 3 界面设计




1. TextInputLayout 是一个非常好用的控件，与 EditText 配合使用可以实现漂亮、简单的动画效果。实现时，只需将 EditText 放在 TextInputLayout 里面即可。


2. 数据的显示采用 RecyclerView 控件，因为这个控件动画效果多。
3. Button 的功能有，添加用户输入的数据到数据库，读取数据库信息到 RecyclerView（刷新），随机生成 1000 条数据并批量加入数据库（有 Insert 方法和 statement 方法），计算耗时。




4. 全部需要文字显示的部分都采用国际化设计。

# 4 代码设计

## 4.1 添加数据到数据库

```
myBtnAdd.setOnClickListener((v) -> {
        if(myEditTextId.getText().toString().equals("") || myEditTextName.getText().toString().equals(""))
            return;
        int id = Integer.valueOf(myEditTextId.getText().toString());
        String letter = myEditTextName.getText().toString();
        SQLiteDatabase db = mySql.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("id", id);
        values.put("letter", letter);
        if(db.insert("data", null, values) == -1)
            Toast.makeText(getApplicationContext(), getString(R.string.alert), Toast.LENGTH_LONG).show();
        else
            Toast.makeText(getApplicationContext(), getString(R.string.added), Toast.LENGTH_LONG).show();
        db.close();
        myEditTextName.setText("");
        myEditTextId.setText("");
});
```

1）首先简单判断用户的输入是否合法，都不为空时，才能继续。本 App 不把设计重心放在输入检测上，而是放在 SQLite 的使用上。

2）通过 SQLiteOpenHelper 获得 Writable 的 SQLiteDataBase。

3）插入操作，第一个参数是表名。根据官方文档的定义，如果插入成功则返回插入的行数，如果失败则返回-1。

```
 * @return the row ID of the newly inserted row, or -1 if an error occurred
```

4）最后记得执行 close()关闭数据库。

### 4.2 从数据库读取数据（刷新）

```
private void notifyList()//Read data from database and show it in RecyclerView
{
    String[] cols = {"id","letter"};
    SQLiteDatabase db = mySql.getReadableDatabase();
    Cursor cursor = db.query("data", cols, null, null, null, null, null, null);
    if(cursor.isAfterLast())
        return;
    cursor.moveToFirst();
    int id = Integer.valueOf(cursor.getString(0));
    String letter = cursor.getString(1);
    idData.add(id);
    nameData.add(letter);
    while(cursor.moveToNext()){
        id = Integer.valueOf(cursor.getString(0));
        letter = cursor.getString(1);
        idData.add(id);
        nameData.add(letter);
    }
    db.close();
    adapter.notifyItemRangeChanged(0, idData.size());
}
```

1）定义一个游标，通过 query 读取到表内的数据。根据官方文档分析 query 的参数，后面的几项（selection, selectionArgs, groupBy, having, orderBy）都为空。

```
* @param table The table name to compile the query against.
* @param columns A list of which columns to return. Passing null will
*            return all columns, which is discouraged to prevent reading
*            data from storage that isn't going to be used.
* @param selection A filter declaring which rows to return, formatted as an
*            SQL WHERE clause (excluding the WHERE itself). Passing null
*            will return all rows for the given table.
* @param selectionArgs You may include ?s in selection, which will be
*         replaced by the values from selectionArgs, in order that they
*         appear in the selection. The values will be bound as Strings.
* @param groupBy A filter declaring how to group rows, formatted as an SQL
*            GROUP BY clause (excluding the GROUP BY itself). Passing null
*            will cause the rows to not be grouped.
* @param having A filter declare which row groups to include in the cursor,
*            if row grouping is being used, formatted as an SQL HAVING
*            clause (excluding the HAVING itself). Passing null will cause
*            all row groups to be included, and is required when row
*            grouping is not being used.
* @param orderBy How to order the rows, formatted as an SQL ORDER BY clause
*            (excluding the ORDER BY itself). Passing null will use the
```

2）Cursor.isAfterLast 是用来判断，游标是否已经达到最后一行的后一个，这里放到一开始是要判断表是否为空。

```
* @return whether the cursor is after the last result.
```

3）Cursor.moveToNext 是将游标移到下一行，如果成功返回 True，失败返回 False。

4）最后要提示 RecyclerView 的适配器更新数据。

## 4.3 Insert 函数批量插入

```
protected void runInsert()
{
    SQLiteDatabase db = mySql.getWritableDatabase();
    db.execSQL("delete from data");
    ContentValues values;
    for(int i = 0; i<sizeOfData; i++)
    {
        values = new ContentValues();
        values.put("id", i);
        values.put("letter", nameData.get(i));
        db.insert("data", null, values);
    }
    db.close();
}
```

1）一开始先执行语句，删除表内所有内容。

2）这种方式是循环 1000 次，每次都通过 insert 函数插入到表里。

## 4.4 使用 SQLite statement 批量插入

```
protected void runInsert2()
{
    SQLiteDatabase db = mySql.getWritableDatabase();
    db.execSQL("delete from data");
    String sql = "insert into data(id, letter)values(?,?)";
    SQLiteStatement stat = db.compileStatement(sql);
    db.beginTransaction();
    for(int i = 0; i<sizeOfData; i++)
    {
        stat.bindLong(1, i);
        stat.bindString(2, nameData.get(i));
        try {
            stat.executeInsert();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
    db.setTransactionSuccessful();
    db.endTransaction();
    db.close();
}
```

1）首先要搞明白 SQLiteStatement 是什么，根据官方文档，它是继承 SQLiteProgram 的一个类，那么 SQLiteProgram 是什么呢？简单的说，理解为一个编译好的 SQLite 程序。beginTransaction 表示开始事务，下面是执行事务的过程，直到 endTransaction。

```
/**
 * Represents a statement that can be executed against a database.  The statement
 * cannot return multiple rows or columns, but single value (1 x 1) result sets
 * are supported.
 * <p>
 * This class is not thread-safe.
 * </p>
 */
public final class SQLiteStatement extends SQLiteProgram {
```

```
/**
 * A base class for compiled SQLite programs.
 * <p>
 * This class is not thread-safe.
 * </p>
 */
public abstract class SQLiteProgram extends SQLiteClosable {
```

2）为 statement 绑定数据。

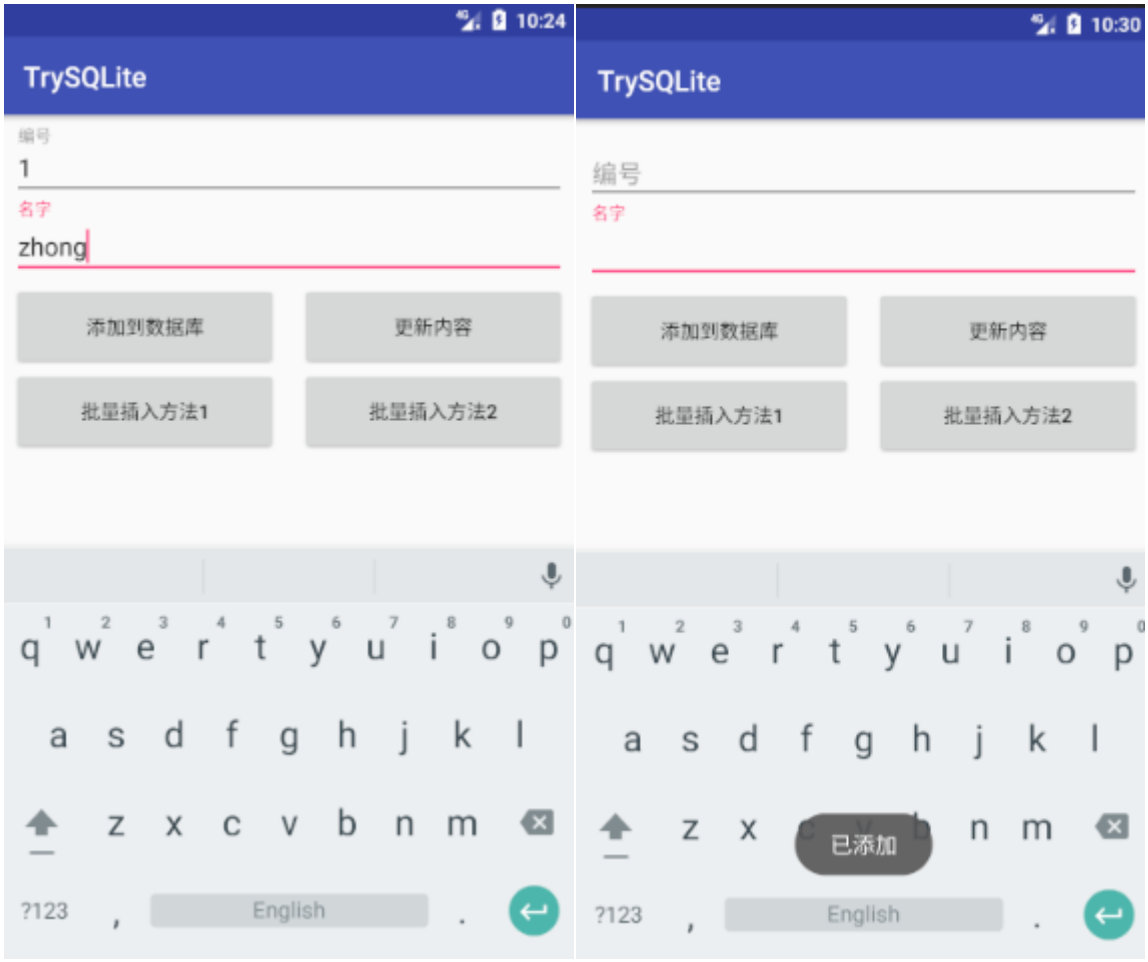
3）执行插入操作。

4）使这个事务执行成功。

```
/**
 * Marks the current transaction as successful. Do not do any more database work between
 * calling this and calling endTransaction. Do as little non-database work as possible in that
 * situation too. If any errors are encountered between this and endTransaction the transaction
 * will still be committed.
 *
 * @throws IllegalStateException if the current thread is not in a transaction or the
 * transaction is already marked as successful.
 */
public void setTransactionSuccessful() {
```

## 5 软件操作流程

基本的添加操作



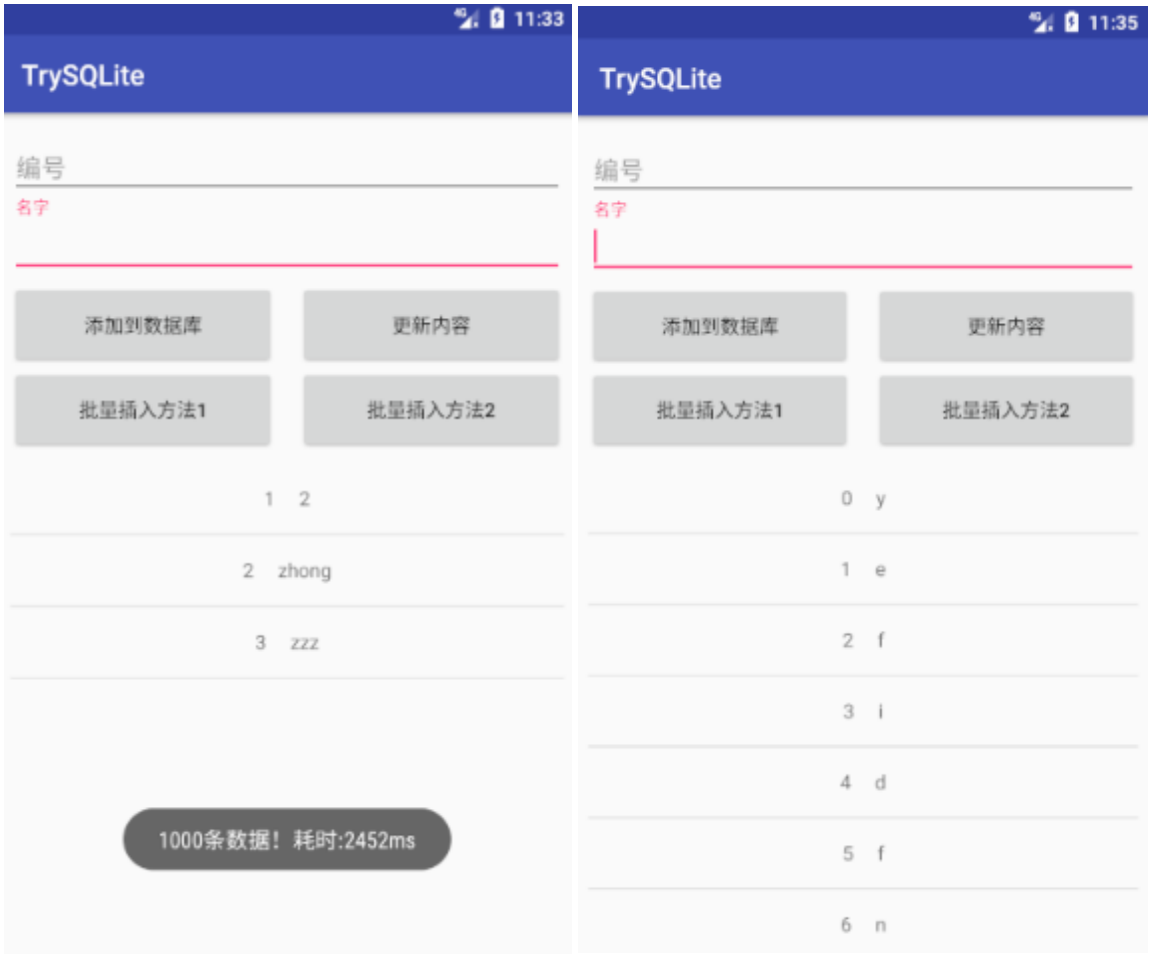

由于 id 是主键，当 id 重复时，会提示“id 已存在”。


10:31
TrySQLite
编号
名字
添加到数据库
更新内容
批量插入方法1
批量插入方法2
id已存在
?123
English

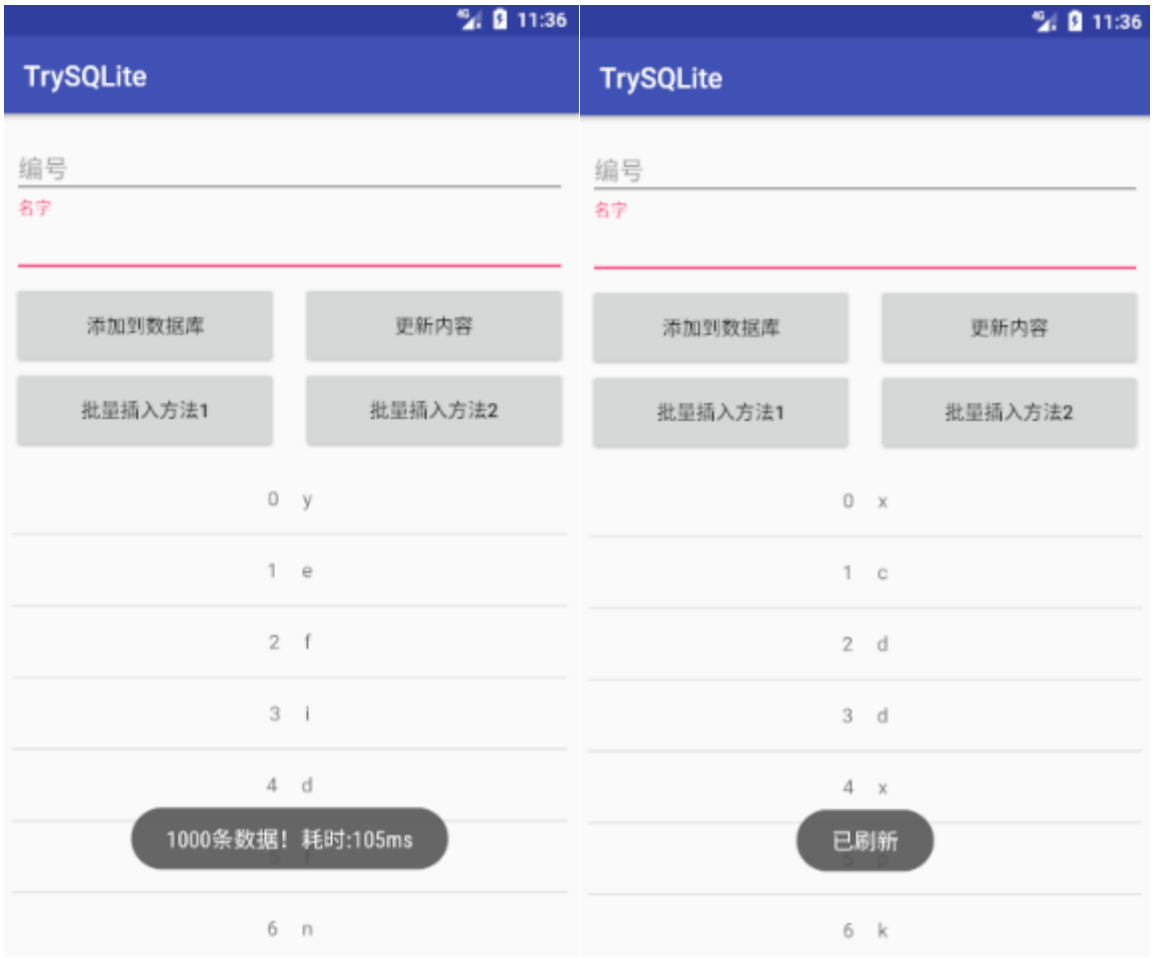
刷新显示内容


10:33
TrySQLite
编号
名字
添加到数据库
更新内容
批量插入方法1
批量插入方法2
1 2
2 zhong
3 zzz
已刷新

批量插入方法 1


11:33
TrySQLite
编号
名字
添加到数据库
更新内容
批量插入方法1
批量插入方法2
1 2
2 zhong
3 zzz
1000条数据！耗时:2452ms
11:35
TrySQLite
编号
名字
添加到数据库
更新内容
批量插入方法1
批量插入方法2
0 y
1 e
2 f
3 l
4 d
5 f
6 n

批量插入方法 2




可以明显看到，第二种方法的速度比第一种快很多。

# 6 难点和解决方案

## 6.1 使用 SQLite statement 批量插入

对这种批量插入方式很陌生，找了网上的代码学习，然后看了官方文档学习内部实现原理。

# 7 不足之处和今后设想

## 7.1 没有单独一个清除数据库按钮

花了比较多的时间在批量插入，没有设置一个清除数据库的按钮。