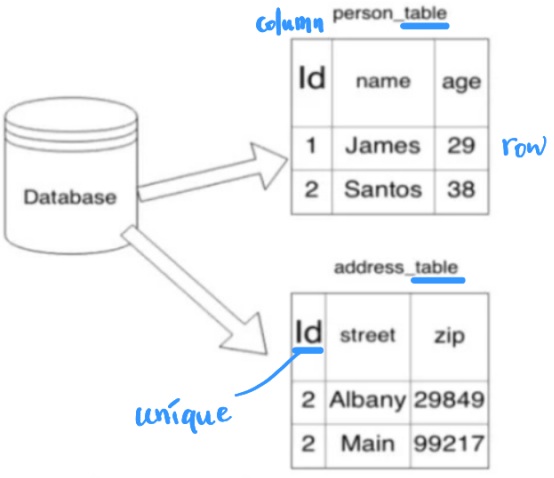
Database in Android

Database - SQLite

Database - a structured way for organizing data into tables.



CRUD

Create

Read

Update

Delete

ContactManager App

1. Create Contact.java in model pkg

*. Contact.java

activity_main.xml | MainActivity.java | Contact.java | DatabaseHandler.java | Util.java

```java
package com.logan.contactmanager.model;

public class Contact {
    private int id;
    private String name;
    private String phoneNumber;

    public Contact() {
    }

    public Contact(int id, String name, String phoneNumber) {
        this.id = id;
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
```

```
28
29      public void setName(String name) { this.name = name; }
32
33      public String getPhoneNumber() { return phoneNumber; }
36
37      public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }
40  }
```

2. Create util/Util.java

activity_main.xml | MainActivity.java | Contact.java | DatabaseHandler.java | Util.java

```
1   package com.logan.contactmanager.util;
2
3   public class Util {
4
5       // Database related itmes
6       public static final int DATABASE_VERSION = 1;
7       public static final String DATABASE_NAME = "contact_db";
8       public static final String TABLE_NAME = "contacts";
9
10      // Contacts table columns
11      public static final String KEY_ID = "id";
12      public static final String KEY_NAME="name";
13      public static final String KEY_PHONE_NUMBER = "phone_number";
14  }
15
```

3. Create data/DataBaseHandler.java

extends SQLiteOpenHelper → 燈泡

→ Implement Methods → OnCreate 和 OnUpgrade

→ 燈泡 → Create Constructor → Select first one → ok

*. Note: ContactManager SQL Table

cursor.getString(0)

cursor 0 cursor 1 cursor 2

db.getContact(1)

id	name	phone_number
id1 → 1	James	124989
id2 → 2	Paul	1636463
⋮	⋮	⋮

*. DataBaseHandler.java

activity_main.xml × MainActivity.java × Contact.java × DatabaseHandler.java × Util.java ×

```java
public class DatabaseHandler extends SQLiteOpenHelper {

    public DatabaseHandler(Context context) {
        super(context, Util.DATABASE_NAME, factory: null, Util.DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // SQL - Structured Query Language
        // create table
        // Watch out "Space"
        String CREATE_CONTACT_TABLE = "CREATE TABLE " + Util.TABLE_NAME + "("
                + Util.KEY_ID + " INTEGER PRIMARY KEY," + Util.KEY_NAME + " TEXT,"
                + Util.KEY_PHONE_NUMBER + " TEXT" + ")";
        db.execSQL(CREATE_CONTACT_TABLE);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Delete old version table
        String DROP_TABLE = "DROP TABLE IF EXISTS";
        //String DROP_TABLE = String.valueOf(R.string.db_drop);
        db.execSQL(DROP_TABLE, new String[]{Util.DATABASE_NAME}); → Delete old table

        // Create new table
        onCreate(db); → create new table
    }

    // CRUD : Create Read Update Delete
    // Add contact
    public void addContact(Contact contact){
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(Util.KEY_NAME, contact.getName());
        values.put(Util.KEY_PHONE_NUMBER, contact.getPhoneNumber());

        // insert to row
        db.insert(Util.TABLE_NAME, nullColumnHack: null, values);
        db.close(); // closing db connection
    }

    // Get contact (setup cursor & transform into a contact obj)
    public Contact getContact(int id){
        SQLiteDatabase db = this.getReadableDatabase();
```

Create Database

C

R

*. 存 data 到 SQL DB

Contact obj → ContentValues obj → 存入 DB

```java
        Cursor cursor = db.query(Util.TABLE_NAME,
                new String[]{Util.KEY_ID, Util.KEY_NAME, Util.KEY_PHONE_NUMBER},
                selection: Util.KEY_ID + "=?", new String[]{String.valueOf(id)},
                groupBy: null, having: null, orderBy: null);

        if (cursor != null){  // not empty
            cursor.moveToFirst();
        }

        Contact contact  = new Contact();
        // contact.setId(cursor.getString(0));
        // cursor.getString : get id, which is a string
        // contact.setId : input int
        contact.setId(Integer.parseInt(cursor.getString( i: 0)));
        contact.setName(cursor.getString( i: 1));
        contact.setPhoneNumber(cursor.getString( i: 2));

        return contact;
    }
    // Get all contact
    public List<Contact> getAllContacts(){
        List<Contact> contactList = new ArrayList<>();
        SQLiteDatabase db = this.getReadableDatabase();

        // Select all contacts
        String selectAll = "SELECT * FROM " + Util.TABLE_NAME;
        Cursor cursor = db.rawQuery(selectAll,  selectionArgs: null);

        // Loop through data
        if (cursor.moveToFirst()) {
            do {
                Contact contact = new Contact();
                contact.setId(Integer.parseInt(cursor.getString( i: 0)));
                contact.setName(cursor.getString( i: 1));
                contact.setPhoneNumber(cursor.getString( i: 2));

                // aff contact obj to list
                contactList.add(contact);
            }while (cursor.moveToNext());
        }

        return contactList;
    }


    // Update contact
    public int updateContact(Contact contact){
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(Util.KEY_NAME, contact.getName());
        values.put(Util.KEY_PHONE_NUMBER, contact.getPhoneNumber());

        // update row
        // update(tablename, values, where id = 43)
        // The return value of the update() method is the number of rows affected in the database.
        return db.update(Util.TABLE_NAME, values,  whereClause: Util.KEY_ID + "=?",
                new String[]{String.valueOf(contact.getId())});
    }

    // Delete single contact in db
    public void deleteContact(Contact contact){
        SQLiteDatabase db = this.getWritableDatabase();

        db.delete(Util.TABLE_NAME,  whereClause: Util.KEY_ID + "=?",
                new String[]{String.valueOf(contact.getId())});

        db.close();
```

利用id 將目標 row 讀到 cursor 中

利用 KEY_ID = 輸入id 去搜尋

再用 cursor 將 data 寫入 contact obj

*. Cursor 的 0, 1, 2 不代表 id 代表 column!

選全部 SQL 寫法

用 do-while loop 每個 row

return 值只是 check 是否 update 成功

```java
    }

    // Get data(contact obj) count
    public int getCount(){
        String countQuery = "SELECT * FROM " + Util.TABLE_NAME;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, selectionArgs: null);

        return cursor.getCount();
    }


}
```

Check dB 中有多少 row

注意：SQL DB 中的資料即使App被関掉仍会存在！

* 統整：

Create : 1° db = getWritableDatabase()

2° db.insert(ContentValue obj)

3° db.close()

Read : 1° db = getReadableDatabase()

read 1 row

2° cursor = db.query(rowid)

3° cursor.getString(column)

cursor 水平移动

read "All"

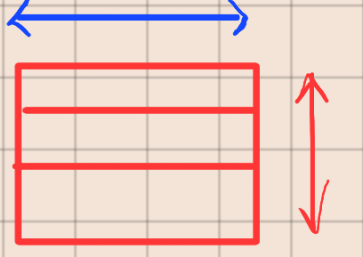
2° cursor = db.rawQuery(....)

3° do-while loop 每个 row

cursor.moveToFirst()

.moveToNext()

Table cursor 垂直移动



Update : 1° db = getWritableDatabase()

2° return db.update (rowid)

Delete : 1° db = getWritableDatabase()

2° db.delete()

3° db.close()