안드로이드 앱의 데이터를 저장하여 영속하는 방법에는 세 가지가 있다: File을 이용하는 방법, SharedPreference를 이용하는 방법, DBMS 프로그램을 사용하는 방법.

● SQLite을 이용한 영속화: File을 이용하는 방법 (Local DB). SQLite(www.sqlite.org)는 오픈 소스로 만들어진 관계형 데이터베이스로 복잡하고 구조화된 애플리케이션 데이터를 저장하고 관리한다. 프로세스가 아닌 라이브러리를 이용하므로 데이터베이스가 애플리케이션의 일부로 통합된다. 즉, 서버가 아니라 애플리케이션에 넣어 사용하는 비교적 가벼운 데이터베이스이다. SQLite을 이용한 데이터는 파일에 저장되며, 다음과 같은 경로에 저장된다.

```
// 앱의 내부 디렉토리 (내장 메모리 공간, file DB)
data/data/[package_name]/databases
```

대부분의 상용 앱들은 서버 데이터베이스에 저장된 데이터를 네트워크 프로그램을 통해 안드로이드에서 이용하는 구조지만, 서버 통신이 안 되는 상황이 발생할 경우를 대비해서 서버에서 받은 데이터를 일정 정도 스마트폰 내부에 저장해야 네트워크 연결이 끊기는 상황에 대응할 수 있다.

▶ SQLiteDatabase 클래스: 데이터베이스 프로그램의 핵심 클래스로 데이터베이스에 데이터를 저장하고 가져오고 수정, 삭제하는 모든 SQL 질의문은 SQLiteDatabase 클래스의 함수를 이용하여 수행한다.

우선 SQLiteDataBase의 객체를 얻어야 한다. openOrCreateDatabase() 함수를 이용해서 객체를 얻는데, 첫 번째 매개변수는 개발자가 지정하는 데이터베이스 파일명이다.

```
SQLiteDatabase db = openOrCreateDatabase("memodb", MODE_PRIVATE, null);
// 첫번째 매개변수: DB 파일명. 테이블이 저장된다.
// 함수명이 openOrCreate - 파일명에 해당하는 파일이 있으면 열고, 없으면 만들고
```

이렇게 얻은 SQLiteDatabase 객체의 함수를 이용해 SQL 구문을 수행한다.

execSQL(String sql)	insert, update, delete, create, drop (select 문이 아닌 나머지) SQL 수행	ex) insert 문 db.execSQL("insert into tb_memo (title, content) values (?,?)", new String[]{title, content});	첫 번째 매개변수가 SQL 문. 이곳에서 데이터 부분(values)을 ?로 작성했다면, 두 번째 매개변수에서 각각의 ?에 대응하는 데이터를 지정한다. 왼쪽 예에서는 ?가 두 개이므로 문자열 두 개를 가지는 배열을 적용했다. 첫 번째 데이터가 첫 번째 ?부분에 적용된다. (데이터 지정을 문자열 배열로. 데이터가 여러 개 지정될 수 있기 때문)
rawQuery(String sql, String[] selectionArgs, Object[] bindArgs)	select SQL 수행. 데이터베이스에 저장된 데이터를 찾아서 가져온다.	Cursor cursor = db.rawQuery("select title, content from	첫 번째 매개변수 SQL문에 ? 표현이 없어 두 번째 매개변수 부분이 null이다. rawQuery() 함수의

		tb_memo order by _id desc limit 1", null); select 어느 컬럼을 from 테이블 order by 소팅 조건 => return cursor (선택된 행의 집합 객체)	결과값은 Cursor 객체이다.

rawQuery() 함수의 결과값은 Cursor 객체로, 선택된 행(row)의 집합 객체이다. 여기서 각 열(column)에 해당하는 데이터를 획득하려면 Cursor 객체를 이용해 행을 선택하고, 선택된 행의 열 데이터를 획득하는 구조이다. Curosr의 행을 선택한 상태여야 열 데이터를 추출할 수 있다.

Cursor의 행을 선택하는 함수다.

moveToNext()	순서상으로 다음 행 선택	while 문 안에
moveToFrist()	가장 첫 번째 행 선택	하나만 선택 (if 문 안에)
moveToLast()	가장 마지막 행 선택	하나만 선택 (if 문 안에)
movetoPrevious()	순서상으로 이전 행 선택	while ans 안에

위의 함수 중 하나를 사용해서 행을 선택하고, 선택된 행의 getString() 함수를 이용하면 해당 행의 원하는 열 데이터를 가져올 수 있다. 이때 getString() 함수의 매개변수는 몇 번째 열의 데이터를 가져오는지 지정한다. 예를 들어, getString(0)이면 첫 번째 열의 데이터를 가져온다.

```
while (cursor.moveToNext()) {
    titleView.setText(cursor.getString(0));
    contentview.setText(cursor.getString(1));
}
```

▶ SQLiteOpenHelper 클래스: 데이터베이스 관리만을 목적으로 하는 클래스. 데이터 저장이나 획득 등의 코드는 SQLiteDatabase 클래스를 이용하여 insert와 select 작업을 하고, 테이블 생성이나 스키마 변경 등의 작업은 SQLiteOpenHelper 클래스로 일원화하는 구조를 사용할 수 있다. 이렇게 함으로서 insert와 select 작업을 수행할 때 테이블 생성 여부를 판단하지 않아도 되는 이점이 있다.

SQLiteOpenHelper는 추상 클래스이므로 서브 클래스를 만들어 사용해야 한다.

```
public classe DBHelper extends SQLiteOpenHelper {
    public static final int DATABASE_VERSION = 1;

    // 생성자: 두 번째 매개변수 "memodb"가 데이터베이스 파일명
    // 네 번째 매개변수 DATABASE_VERSION은 데이터베이스 버전 정보
    public DBHelper(Context context) {
        super(context, "memodb", null, DATABASE_VERSION);
    }

    // 앱이 설치되어 SQLiteOpenHelper 클래스가 최초로 사용되는 순간 호출.
    // 앱에서 가장 처음 한 번만 수행되는 코드가 작성되는 함수로, 대부분 여기서 테이블을 생성.
    @Override
    public void onCreate(SQLitDatabase db) {
        // ...
    }

    // SQLiteOpenHelper 클래스의 생성자에 전달되는 데이터베이스 버전 정보가 변경될 때마다
    // 반복해서 호출. 즉, 테이블의 스키마 부분을 변경하기 위한 용도로 사용된다.
    // 단, version이 변경되지 않으면 호출되지 않는다.
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // ...
    }
}
```

이렇게 SQLiteOpenHelper을 정의했다면 실제 SQL문 수행을 위해 SQLiteDatabase 객체를 핼퍼 클래스를 통해서 획득한다. 이때는 SQLiteOpenHelper 클래스의 getReaderableDatabase() 함수와 getWritableDatabase() 함수를 이용한다.

```
DBHelper helper = new DBHelper(this);
SQLiteDatabase db = helper.getWritableDatabase();
```

▶ insert(), query(), update(), delete() 함수 이용: SQLiteDatabase 클래스의 함수를 통한 SQL 문 수행 시에 rawQuery9) 함수와 execSQL() 함수 이외에 다음 함수를 이용할 수도 있다.

insert(String table, String nullColumnHack, Contentvalues values) // table 명, ___ , insert하기 위한 데이터 (map 객체: key-value인데 key가 column명)
update(String table, Contentvalues values, String whereClause, String[] whereArgs) // table 명, 업데이트 하기 위한 테이블 명, where 조건, ___
delete(String table, String whereClause, String[] whereArgs)
query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) // table 명, 획득하고자 하는 column들, where 조건 (? 데이터), ?에 해당되는 데이터(문자열 배열), group by 조건, ...

rawQuery() 함수와 execSQL() 함수는 개발자가 직접 SQL 문을 매개변수로 주어야 하지만, 위의 함수들을 SQL 문을 만들기 위한 정보만 매개변수로 주면, 자동으로 SQL 문을 만들어 실행해 준다. 이 중 insert()와 update() 함수는 ContentValues 클래스를 매개변수로 넘기는데, 이 클래스는 insert 데이터와 update 데이터를 표현하는 집합 객체이다. key-value 형식으로 이용되며 key 값이 실제 테이블의 열(column)이다.

```
// Ex) USER TB 테이블에 values 객체의 내용을 삽입하는 코드

ContentValues values = new ContentValues();
values.put("name", "kkang");
values.put("phone", "0100000");
db.insert("USER_TB", null, values);
```

```
// ex) query() 함수를 이용해 USER_TB 테이블에서 ID가 "kkang"인 사용자를 선택하여
// 해당 사용자의 name과 phone값을 불러오는 코드. group by, having, order by 조건은
// null을 대입하여 생략.

Cursor c = db.query("USER_TB", new String[]{"name", "phone"},
           "ID=?", new String[]{"kkang"},
           null, null, null);
```

```java
// New > Java Class 자바 클래스 파일 만들기
package com.android.practicesqlite;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 1;

    public DBHelper(Context context) {
        // 두 번째 매개 변수: 데이터베이스 파일명
        super(context, "memodb", null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String memoSQL = "create table tb_memo " + "(_id integer primary key autoincreme
                         "title," + "content)";
        // 테이블 생성
        // 칼럼은 _id, title, content 3개로 구성
        db.execSQL(memoSQL);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // 만약 테이블 생성을 잘못해서 수정해도 onCreate() 함수는 앱 설치 후 한 번만 호출되므
        // 수정한 부분이 반영되지 않는다. 이럴 때 DATABASE_VERSION의 값이 증가하면 onUpgrade
        // 자동으로 호출되고, 잘못 만든 테이블을 삭제 후 다시 만든다.
        if (newVersion == DATABASE_VERSION) {
            // 잘못 만든 테이블 삭제
            db.execSQL("drop table tb_memo");
            // 테이블 다시 만들기
            onCreate(db);
        }
    }
}
```

```java
//사용자 입력 데이터를 테이블에 저장하기 위한 MainActivity.java 파일 작성
package com.android.practicesqlite;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;

import com.android.practicesqlite.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        binding.addButton.setOnClickListener(view -> {
            // 입력받은 제목과 내용을 문자열로 전환하여 String 변수에 할당받기
            String title = binding.title.getText().toString();
            String content = binding.content.getText().toString();

            // DBHelper 객체를 통해 SQLiteDatabase 객체 획득
            DBHelper helper = new DBHelper(this);
            SQLiteDatabase db = helper.getWritableDatabase();

            // insert SQL문 - 테이블에 제목과 내용 삽입하기 (title, content 콜럼에 values t
            db.execSQL("insert into tb_memo (title, content) values (?,?)",
                    new String[]{title, content});
            // 자바 소스 코드에서 SQL 문을 표현할 때 데이터가 들어갈 자리를 ?로 표현한 후
            // 이후 매개변수에서 ?에 해당하는 데이터를 지정하는 방식은 소프트웨어 개발에서 흔히

            db.close();

            // 테이블에 데이터 저장 후에 결과를 확인할 액티비티 ReadDBActivity로 화면 전환
            Intent intent = new Intent(this, ReadDBActivity.class);
            startActivity(intent);
        });
    }
}
```

```java
// 테이블에 저장된 데이터를 선택해서 화면에 뿌리기 위한 ReadDBActivity.java
package com.android.practicesqlite;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.android.practicesqlite.databinding.ActivityReadDbactivityBinding;

public class ReadDBActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ActivityReadDbactivityBinding binding = ActivityReadDbactivityBinding.inflate(ge
        setContentView(binding.getRoot());

        // DBHelper 객체를 통해 SQLiteDatabase 객체 획득
        DBHelper helper = new DBHelper(this);
        SQLiteDatabase db = helper.getWritableDatabase();

        // select SQL 구문: title, content column 선택 from tb_memo table order by 소팅 ㄱ
        Cursor cursor = db.rawQuery("select title, content from tb_memo order by _id des

        // Cursor 객체를 통해 행을 선택하고 선택한 행의 열 데이터를 획득해서 화면에 띄우기기
        while (cursor.moveToNext()) {
            binding.readTitle.setText(cursor.getString(0));
            binding.readContent.setText(cursor.getString(1));
        }

        db.close();
    }
}
```

```
// activity_main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">


    <EditText
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="제목을 입력하세요"/>


    <EditText
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:inputType="textMultiLine"
        android:scrollbars="vertical"
        android:hint="메모를 입력하세요"/>


    <Button
        android:id="@+id/add_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ADD" />


</LinearLayout>
```

```
// activity_read_dbactivity.xml

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="DB Select 결과" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Title : "/>
        <TextView
            android:id="@+id/read_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Content : "/>
        <TextView
            android:id="@+id/read_content"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>

</LinearLayout>
```

Resources: 깡쌤의 안드로이드 프로그래밍