CS5530
Mobile/Wireless Systems
Android Databases

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

cat announce.txt_

- Project extension
  - Midnight, May 10 the latest
  - I need to provide grade 72 hours after final exam
- Plan for today: database + final review
- Project demo on Monday

Database on Android

- SQLite
  - Requires limited memory at runtime (approx. 250 KByte)
    - Good candidate from being embedded into other runtimes
  - Supports data types TEXT, INTEGER and REAL
    - All other types must be converted into one of these fields before getting saved
    - SQLite does not validate if the types written to the columns are actually of the defined type

Table Name: Contacts

Field	Type	Key
id	INT	PRI
name	TEXT	
phone_number	TEXT	

Database on Android

- SQLite is embedded into every Android device
  - Using an SQLite database in Android does not require a setup procedure or administration of the database
  - If your app creates a database, this database is by default in directory DATA/data/APP_NAME/databases/FILENAME (needs root to access)
  - Package to import: android.database.sqlite

SQLiteOpenHelper

- To create and upgrade a database in app: create a subclass of SQLiteOpenHelper
  - In the constructor, call super() method of SQLiteOpenHelper, specifying database name and current database version
    - super(context, DATABASE_NAME, null, DATABASE_VERSION);

SQLiteOpenHelper

- To create and upgrade a database in app: create a subclass of SQLiteOpenHelper
  - In the constructor, call super() method of SQLiteOpenHelper, specifying database name and current database version
  - Override methods to create and update database:
    - onCreate() is called automatically when the database is accessed but no tables have been created
    - onUpgrade() called, if the database version is increased. This method allows updating an existing database schema, or to drop the existing database and recreate it via the onCreate() method.

SQLiteOpenHelper

- Example
  - ```
    @Override
    public void onCreate(SQLiteDatabase db) {
       String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
             + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
             + KEY_PH_NO + " TEXT" + ")";
       db.execSQL(CREATE_CONTACTS_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
       // Drop older table if existed
       db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
       // Create tables again
       onCreate(db);
    }
    ```

```
Table Name: Contacts

+------------------+--------------+------+
| Field            | Type         | Key  |
+------------------+--------------+------+
| id               | INT          | PRI  |
| name             | TEXT         |      |
| phone_number     | TEXT         |      |
+------------------+--------------+------+
```
```

SQLiteOpenHelper

- **SQLiteOpenHelper** provides `getReadableDatabase()` and `getWriteableDatabase()` to get access to an `SQLiteDatabase` object
- **`SQLiteDatabase`** provides methods to open, query, update and close the database
  - More specifically the insert(), update() and delete() methods
  - Also provides the execSQL() method, which allows to execute an SQL statement directly

ContentValues

- Insert example
  - db.insert(TABLE_CONTACTS, null, values);
- execSQL example
  - db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
- The object ContentValues allows to define key/values
  - The key represents the table column identifier and the value represents the content for this column
  - ContentValues can be used for inserts and updates of database entries

Adding Record

- Example
  - void addContact(Contact contact) {

```
SQLiteDatabase db = this.getWritableDatabase();

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

// Inserting Row
db.insert(TABLE_CONTACTS, null, values);
db.close(); // Closing database connection
}
```

Query

- Queries can be created via rawQuery() and query() methods
  - rawQuery() directly accepts an SQL select statement as input
    - Syntax: rawQuery(String sql, String [] selectionArgs)
    - Placeholder values in the where clause via ?: you pass them as the selectionArgs parameter to the query
    - Cursor cursor = db.rawQuery("select * from todo where _id = ?", new String[] {id});

Query

- Queries can be created via rawQuery() and query() methods
  - rawQuery() directly accepts an SQL select statement as input
    - Syntax: rawQuery(String sql, String [] selectionArgs)
    - Placeholder values in the where clause via ?: you pass them as the selectionArgs parameter to the query
    - Cursor cursor = db.rawQuery("select * from todo where _id = ?", new String[] {id});
  - query() provides a structured interface for specifying the SQL query
    - Cursor cursor = db.query(tableName, tableColumns, whereClause, whereArgs, groupBy, having, orderBy);
    - Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
      KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
      new String[] { String.valueOf(id) }, null, null, null, null);

Query

- Cursor
  - A query returns a Cursor object
  - A Cursor represents the result of a query and points to one row of the query result
    - Buffer the query results efficiently; as it does not have to load all data into memory
  - To get the number of elements of the resulting query, use getCount()
  - To move between individual data rows, use moveToFirst() and moveToNext()

Query

- Cursor
  - List<Contact> contactList = new ArrayList<Contact>();
    String selectQuery = "SELECT * FROM " + TABLE_CONTACTS;
    Cursor cursor = this.getWritableDatabase().rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (**cursor.moveToFirst()**) {
      do {
        Contact contact = new Contact();
        contact.setID(Integer.parseInt(cursor.getString(0)));
        contact.setName(cursor.getString(1));
        contact.setPhoneNumber(cursor.getString(2));
        contactList.add(contact);
      } while (**cursor.moveToNext()**);
    }