Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

# Data Storage Techniques

## Study Guide

**Prof. Dr. Shiv Shakti Shrivastava**
CSE, PIT
Parul University

1. Data Storage Techniques: Shared Preferences,
2. Files and Directories,
3. SQLite Database Connectivity and Operations,
4. Content Providers: Basics,
5. Content URI, Content Resolver,
6. Built-in content providers.

# Data Storage Techniques in Android

Android offers several mechanisms for data storage to accommodate different needs such as saving preferences, files, or large datasets:

**Overview Table: Storage Options**

| Storage Type | Use Case | Access Scope | Example Usage |
|---|---|---|---|
| Shared Preferences | App settings, user preferences | App-only | User login state |
| Files (Internal) | App files, small data sets | App-only | User profile image |
| Files (External) | Media, documents | Shared/External | Photos, downloads |
| SQLite Database | Structured, relational data | App-only | Contacts, records |
| Content Providers | Sharing data between apps | Inter-app | Contacts, calendar |

---

**Shared Preferences**

- **Purpose:** Store primitive data as key-value pairs (String, int, boolean, etc.) suitable for user or app settings.
- **How it works:** Data is private to the app, stored in XML files.

**Code Example:**

java

```
// Save data in Shared Preferences
SharedPreferences sharedPref = getSharedPreferences("MyPrefs", MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putString("username", "user123");
editor.apply();

// Retrieve data
String username = sharedPref.getString("username", "default");
```

- **Use Cases:** Remembering login, theme preferences, feature switches.swiftorial+1

---

**Files and Directories**

**Internal Storage**

- Files are protected and accessible only by the app.
- Best for personal or sensitive app data.

**Sample Code:**

java

```
String fileName = "sample.txt";
String fileContents = "Hello, World!";
try (FileOutputStream fos = openFileOutput(fileName, Context.MODE_PRIVATE)) {
    fos.write(fileContents.getBytes());
}
```

**External Storage**

- Files stored here are public and can be accessed by other apps if permissions are granted.
- Permissions required (READ/WRITE_EXTERNAL_STORAGE).

**Sample Code:**

java

```
File file = new File(Environment.getExternalStorageDirectory(), "sample.txt");
try (FileOutputStream fos = new FileOutputStream(file)) {
    fos.write("Hello, World!".getBytes());
}
```

- **Security Note:** Sensitive data should not be stored here due to its public nature.ccecc.acm+1

---

**SQLite Database Connectivity and Operations**

- **Best for:** Structured data, offline storage, queries, and multi-row datasets.
- **Implementation:** SQLiteOpenHelper class manages database creation and version management.

**Key Operations Table:**

| Operation | Method | Example |
|---|---|---|
| Create/Open DB | getWritableDatabase() | DB initialization |
| Insert | db.insert() | Insert new row |
| Query | db.query() | rawQuery() |
| Update | db.update() | Modify existing records |
| Delete | db.delete() | Remove rows |

**Sample SQLite Helper (Java):**

java

```
public class MyDBHelper extends SQLiteOpenHelper {
  public MyDBHelper(Context context) {
    super(context, "MyDatabase", null, 1);
  }

  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT)");
  }
  public void onUpgrade(SQLiteDatabase db, int oldVer, int newVer) {
    db.execSQL("DROP TABLE IF EXISTS users");
    onCreate(db);
  }
}
```

**CRUD Illustration Figure (Conceptual):**

- (Imagine a table with arrows for Create, Read, Update, and Delete operations acting on rows in the database).mines.humanoriented+2

---

**Content Providers: Architecture and Usage**

**Basics**

- **Purpose:** Allow sharing of data between applications in a secure and consistent way.
- **Structure:** Data is exposed in a tabular form, similar to SQL tables.
- **Key Components:** ContentProvider, ContentResolver, Content URI.stuff.mit+2

**Content URI**

- Unique address required to access data from a content provider, e.g., content://com.example.provider/table.

**ContentResolver**

- Client-side API to perform CRUD (Create, Read, Update, Delete) operations on a content provider via URIs.

**Example Table: Built-in User Dictionary Content Provider**

| word | app id | frequency | locale | _ID |
|---|---|---|---|---|
| mapreduce | user1 | 100 | en_US | 1 |
| precompiler | user14 | 200 | fr_FR | 2 |
| applet | user2 | 225 | fr_CA | 3 |

**CRUD with ContentResolver (Java sample):**

java

```
// Querying a content provider
Cursor cursor = getContentResolver().query(
   ContactsContract.Contacts.CONTENT_URI, null, null, null, null
);
while(cursor.moveToNext()){
   String name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
}
cursor.close();
```

**Built-in Content Providers**

- **ContactsContract**: Manages user's contacts
- **MediaStore**: Manages multimedia files
- **CallLog**: Accesses call history
- **CalendarContract**: For calendar events
- **UserDictionary**: Stores user-added words

**List of Some ContentProvider URIs:**

| Provider | Content URI Example |
|---|---|
| Contacts | content://contacts/people |
| UserDictionary | content://user_dictionary/words |
| MediaStore | content://media/external/images/media |

**Conceptual Chart (Textual Description):**

- App-1 (ContentProvider) ←→ ContentResolver ←→ App-2 (Consumer)
- Each side deals with data as rows/tables, secured and abstracted by the provider interface.suwish+2

---

**Further Resources**

- Android Developers: Powerful official documentation for each topic (see references inside most technical university materials)
- Tutorials and PDFs linked in study guides offer step-by-step code and figures for classroom and self-study.codezup+2

---

This material covers all critical aspects (concepts, code, security, tables, and architecture) to support deep learning and instructional use for both classroom and exam contexts, referencing examples, sample code, and tabular figures from authoritative sources.slideshare+6

1. https://www.swiftorial.com/swiftlessons/mobile-app-development/android-development/data-storage-in-android
2. https://codezup.com/android-data-storage-guide/
3. http://ccecc.acm.org/files/publications/Lab-6-Data-Storage-in-Android.pdf
4. http://mines.humanoriented.com/classes/2011/spring/csci403/books/sqlite01.pdf
5. https://www.slideshare.net/slideshow/databases-with-sqlite3pdf/252131601
6. https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/providers/content-provider-basics.html
7. http://api.suwish.com/android/guide/topics/providers/content-provider-basics.html
8. https://www.cs.utexas.edu/~scottm/cs371m/Handouts/Slides/21_Content_Providers.pdf
9. https://it.aduacademy.in/Android/chapter6/Android%20Techniques%20for%20Saving%20Data.pdf
10. https://www.studocu.com/row/document/arba-minch-university/mobile-app-development/chapter-5-mp/109134693
11. https://www.androidengineers.in/roadmap/data-storage
12. https://www.dre.vanderbilt.edu/~schmidt/cs282/PDFs/9-Content-Providers.pdf
13. https://programmershouse.ir/Library/Files/SQLite.pdf
14. https://www.slideshare.net/malikid/android-data-storage
15. https://developer.android.com/guide/topics/providers/content-provider-basics

16. https://www.sqlite.org/capi3ref.html
17. https://www.egyankosh.ac.in/bitstream/123456789/76227/1/Unit-11.pdf
18. https://www.wideskills.com/android/application-components/content-providers
19. https://es.scribd.com/document/513977836/Give-a-Try-Database-Connectivity
20. https://www.vskills.in/certification/certified-android-apps-developer-data-storage-in-android

Parul® University
Vadodara, Gujarat

NAAC GRADE A++


https://paruluniversity.ac.in/