# １１　SQLiteの概要

Androidにはリレーショナルデータベースである SQLite が標準で掲載されています。リレーショナルデータベースは、データを表の形で扱うことができるデータベースです。

リレーショナルデータベースには、ＳＱＬと呼ばれる言語によって簡単にデータの操作や問い合わせができようになっています。

SQLiteは、クライアントサーバ形式ではなく端末の中で処理が完結します。また、取り扱いがシンプルで、サーバのホスト名やポート番号、ログイン ID、パスワードの指定も不要です。保存先は、「data/data/パッケージ名/databases/ファイル名」となります。

## 11.1　SQLiteDatabaseオブジェクト

Android では、SQLiteDatabase オブジェクトを使って、データベースを操作します。SQLiteDatabase オブジェクトを取得するには、通常、ヘルパークラスとして、SQLiteOpenHelper クラスを継承したクラスを作成して、このクラスより SQLiteDatabase オブジェクトを取得します。

データベースを使用するには、まずデータベースを保存するファイルを作成し、次にデータを格納するテーブルを定義し……といった準備が必要になります。SQLiteOpenHelper クラスを使用することで、こうした煩わしい初期処理を自動化できます。

【ヘルパークラス】
ヘルパークラスは、一般に、アプリケーションによって特定のタスクを実行するために使用されるユーティリティです。通常、アプリケーション全体で何度も実行される共通のタスクのロジックを集中化するこれらのクラスが作成されます。

## 11.2　SQLiteOpenHelper クラスでデータベースの生成

MyOpenHelper.java では、「SQLiteOpenHelper」というクラスを継承して「MyOpenHelper」というクラスが作成されています。これがヘルパークラスです。

「SQLiteOpenHelper」は、抽象クラスであり、データベースの作成やバージョン管理に役立つヘルパークラスです。データベースの作成がまだ行われていなければ作成を行い、作成されていればそれを開きます。また、必要とあればデータベースのアップグレードを行います。

SQLiteOpenHelper では、４つの引数を持つコンストラクタが用意されています。Context、データベース名、CursorFactory、データベースのバージョン番号といった値が渡されます。コンストラクタを定義する際には、この引数が４つあるスーパークラスのコンストラクタを呼び出す必要がある、という点に注意しましょう。

SQLiteOpenHelper にはいくつかのメソッドがありますが、一般的にはオブジェクトが生成される際の「onCreate」と、データベースの更新時に呼び出される「onUpgrade」は用意しておく必要があります。SQLiteOpenHelper クラスは抽象クラスのため、利用する場合はそれを継承した独自クラスを定義し、抽象メソッドである onCreate()メソッドや onUpgrade()メソッドなどをオーバーライドして実装します。

いずれも「SQLiteDatabase」というクラスのインスタンスが引数に渡されています。これは、SQLite にアクセスするための機能を提供するクラスです。データベースアクセスは、この SQLiteDatabase にあるメソッドを呼び出して行います。

### 11.2.1 データベースヘルパーの定義

```
private static class MyOpenHelper extends SQLiteOpenHelper {
    //■データベースヘルパーのコンストラクタ
    public MyOpenHelper(Context context)
    {
```

```
        super(context, "CustomerCard.db", null, DB_VERSION);
    }
```

SQLiteOpenHelper クラスを継承したクラス「**MyOpenHelper**」を定義し、それを利用します。

| オーバーライドするメソッド | メソッドの呼び出し |
|---|---|
| onCreate()メソッド | データベース生成時に呼ばれる |
| onUpgrade()メソッド | データベースアップグレード時に呼ばれる |

### 11.2.2 データベースヘルパーのコンストラクタ

super(context,**データベースファイル名**,**ファクトリー**,**バージョン**);

## 11.3 SQLiteの主な命令

| 命令 | 説明 |
|---|---|
| CREATE | テーブルの生成 |
| DROP | テーブルの削除 |
| SELECT | レコードの検索及び、データの抽出 |
| UPDATE | レコードの更新 |
| INSERT | レコードの挿入 |
| DELETE | レコードの削除 |

## 11.4 テーブルの作成

```
//■データベースの生成
public void onCreate(SQLiteDatabase db)
{
    String strSQL = "CREATE TABLE product" + "(id INTEGER PRIMARY KEY,
                            name STRING, address STRING, shoeSize double)";
    db.execSQL(strSQL);
}
```

文法

テーブルの生成

CREATE TABLE テーブル名 (カラム名 1, カラム名 2, ...);

SQL 命令を実行するには、SQLiteDatabase クラスの execSQL()メソッドを使います。

カラムに対して PRIMARY KEY 制約を設定することができますが、そのカラムのデータ型が INTEGER だった場合には自動的に連続する数値が格納されるようになります。

テーブルにデータを追加するときに、対象のカラムに値を代入しないと、そのカラムに格納されている最大の値に 1 を加えた値がカラムの値として格納されます。

カラムのデータ型は「INT」が含まれる場合、全て INTEGER 型となりますが、連番が自動的に割り振られるのはカラムに対して「INTEGER PRIMARY KEY」と記述した場合だけです。「INT PRIMARY KEY」ではこのような特別な動作はしません。

### 11.4.1 テーブルの列に指定できる型

| 型 | 説明 |
|---|---|
| TEXT | 文字列を格納する。UTF-8, UTF-16BE or UTF-16-LE のいずれかで格納 |
| NUMERIC | 整数または浮動小数点数を格納する。 |
| INTEGER | 符号付整数を格納する。1, 2, 3, 4, 6, or 8 バイトで格納 |
| REAL | 浮動小数点数を格納する。8 バイトで格納 |
| BLOB | Binary Large Object。入力データをそのまま格納 |

| NONE | 変換なし |
|---|---|

### 11.4.2 テーブルの列に設定できる制約

| 制約 | 説明 |
|---|---|
| PRIMARY KEY | テーブルの主キー |
| NOT NULL | NULL を保存するとエラーになる |
| DEFAULT | 生成時に指定されない場合のデフォルト値を設定 |
| UNIQUE | 重複した値を格納した場合はエラーになる |
| AUTOINCREMENT | 生成時に自動で値をナンバリングする |

### 11.4.3 PRIMARY KEY と UNIQUE の相違点

①PRIMARY KEY はテーブル定義時に一度だけ設定することができるが、UNIQUE は他の制約と同じ扱いなので、定義時以外にもテーブルの変更でも設定できる。

②PRIMARY KEY は設定した後に値を書き換えられないが、UNIQUE は書き換えられる。

③PRIMARY KEY は値に NULL を設定できないが、UNIQUE には NULL を設定できる。

## 11.5 データベースのアップグレード

```
//■データベースのアップグレード
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    db.execSQL("drop talbe if exists "+DB_TABLE);
    onCreate(db);
}
```

文法

テーブルの削除

DROP TABLE テーブル名;

## 11.6 データベースオブジェクトの取得

```
//■データベースオブジェクトの取得
MyOpenHelper helper = new MyOpenHelper(this);
db = helper.getWritableDatabase();
```

データベースヘルパーを利用して、SQLiteDatabase 型のデータベースオブジェクトを取得します。具体的には、MyOpenHelper オブジェクトを生成後、getWritableDatabase()メソッドを呼びます。

SQLiteDatabase オブジェクトを取得するのに、getWritableDatabase メソッドを使用していますが、 SQLiteDatabase オブジェクトを取得するには、getReadableDatabase メソッドもあります。

getReadableDatabase メソッドと getWritableDatabase メソッドの違いは、getReadableDatabase メソッドが読込み用にデータベースをオープンするのに対して、getWritableDatabase メソッドは読み書き用にデータベースをオープンすることです。

getReadableDatabase メソッドや getWritableDatabase メソッドを使ってオープンしたデータベースは、close メソッドを使ってデータベースをクローズする事を忘れないようにして下さい。

## 11.7 テーブルへの新規書き込み

```
//■テーブルへの新規書き込み
strSQL = "INSERT INTO product(id, name, address, shoeSize)" +
                          " VALUES(" + id + ",'" + name + "','" +
```

```
                                                address + "','" + shoeSize + "')";
db.execSQL(strSQL);
```

文法

データの生成

INSERT INTO テーブル名 (列名, 列名, ...) VALUE (データ, データ, ...);

## 11.8 テーブルの書き換え

```
//■テーブルの書き換え
strSQL = "UPDATE product SET name = '" + name + "', address = '" + address +
                                                "', shoeSize = '" + shoeSize +
                      "' WHERE id = '" +  id + "'";
db.execSQL(strSQL);
```

文法

データの更新

UPDATE テーブル名 SET 列名 = データ, 列名 = データ, ... [WHERE] 条件;

## 11.9 テーブルのレコードの削除

```
//■テーブルのレコードの削除
strSQL = "DELETE FROM product WHERE id = '" +  id + "'";
db.execSQL(strSQL);
```

文法

データの削除

DELETE FROM テーブル名 [WHERE] 条件;

## 11.10 データの検索

```
//■データの検索
strSQL = "SELECT * FROM product WHERE id = '" + id + "'";
Cursor cr = db.rawQuery(strSQL, null);
```

文法

データの検索

SELECT 列名, ... FROM テーブル名, ...

[WHERE 条件式]

[GROUP BY 列名 [HAVING 条件式 ]]

[ORDER BY 列名...]

・検索結果は、Cursor を使って取得します。Cursor は可変長の List のようなクラスです。

# １２ Android エミュレータで日本語入力をする方法

EditText などに日本語を入力する場合、次のような設定が必要です。

## 12.1 ひとまずAndroidエミュレータを日本語表示にする

①Android エミュレータの「MENU」⇒「Settings」⇒「Language & keybord」を選択

②「Select language」から「日本語」を選択

これでエミュレータの表示が全て日本語になります。

## 12.2　日本語入力を使う

①同じく「MENU」⇒「設定」⇒「言語とキーボード」を選択

②「Japanese IME」にチェックを入れる

これで日本語が入力できます。