第5章 数据管理

课程思政 5

本章要点

- 关系数据库的使用
- 分布式数据服务

本章知识结构图（见图 5-1）

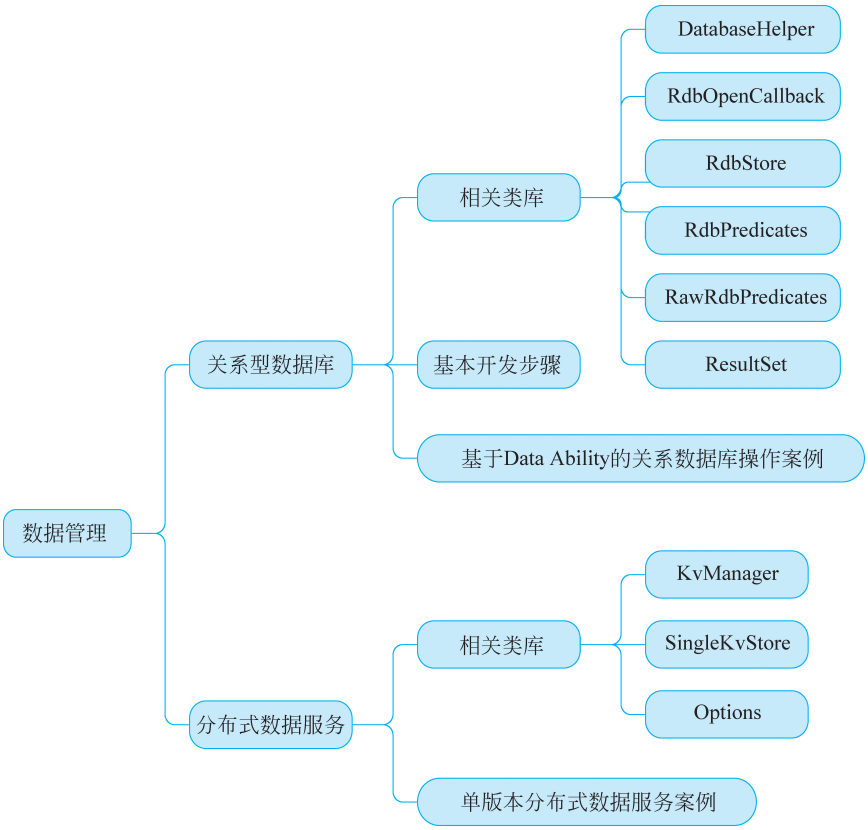

图 5-1　本章知识结构图

本章示例(见图 5-2)

图 5-2　本章示例图

一个较好的应用程序,应该能够为用户保存个性化的设置,能够保存用户的使用记录,而这些都离不开数据的存储,HarmonyOS 提供了关系数据库来存储信息。5.1.5 节将通过一个基于 Data Ability 的关系数据库操作案例详细讲解关系数据库的使用方式,读者可以根据该案例再次开发出其他更为复杂的应用程序。

HarmonyOS 还为开发者提供了分布式数据服务,当开发者想在两台或几台设备上共享同一数据时,就可以通过使用分布式数据服务来完成,5.2.4 节将通过一个分布式数据服务案例详细讲解 HarmonyOS 的分布式数据服务的使用方式。

5.1　关系数据库

5.1.1　关系数据库介绍

关系数据库(Relational Database,RDB)是指采用关系模型组织数据的数据库,是创建在关系模型基础上的数据库,以行和列的形式存储数据,方便用户理解。关系数据库可以理解为由二维表及其之间的关系组成的一个数据组织。

HarmonyOS 关系数据库基于 SQLite 组件实现,提供了一套完整的对本地数据库进行管理的机制,对外提供了一系列的增加、删除、修改、查询接口,也可以直接运行开发者输入的 SQL 语句来满足复杂的场景需要。

5.1.2 约束与限制

- 数据库中连接池的最大数量是 4 个，用以管理用户的读写操作。
- 为保证数据的准确性，数据库同一时间只能支持一个写操作。

5.1.3 关系数据库相关类

为了操作和管理关系数据库，HarmonyOS 为开发者提供了一些相关类，常用的有 DatabaseHelper、RdbOpenCallback、RdbStore、RdbPredicates、RawRdbPredicates、ResultSet。

DatabaseHelper 是 HarmonyOS 提供的管理数据库的工具类，主要用于关系数据库的创建、打开和删除。DatabaseHelper 类定义的常用操作方法如表 5-1 所示。

表 5-1 DatabaseHelper 类定义的常用操作方法

方 法	描 述
publicRdbStore getRdbStore(StoreConfig config, int version, RdbOpenCallback openCallback, ResultSetHook resultSetHook)	根据配置创建或打开数据库
public booleandeleteRdbStore(String name)	删除指定的数据库

RdbOpenCallback 是 HarmonyOS 为开发者提供的管理数据库创建、升级和降级的工具类。开发者可以创建一个子类来实现 onCreate()和 onUpgrade()等方法，如果一个数据库已经存在，它将被打开；如果不存在数据库，将创建一个新数据库。在数据库升级过程中，也会调用此类的方法。RdbOpenCallback 类定义的常用操作方法如表 5-2 所示。

表 5-2 RdbOpenCallback 类定义的常用操作方法

方 法	描 述
public abstract void onCreate(RdbStore store)	数据库创建时被调用，开发者可以在该方法中初始化表结构，并添加一些应用使用到的初始化数据
public abstract voidonUpgrade(RdbStore store, int currentVersion, int targetVersion)	数据库需要升级时被回调
void onOpen(RdbStore store)	打开数据库时调用

RdbStore 是 HarmonyOS 提供的用来对关系数据库进行增加、删除、修改、查询操作的工具类。RdbStore 类定义的常用操作方法如表 5-3 所示。

表 5-3 RdbStore 类定义的常用操作方法

方 法	描 述
long insert(String table,ValuesBucket initialValues)	向数据库插入数据
int update(ValuesBucket values, AbsRdbPredicates predicates)	更新数据库表中符合谓词指定条件的数据
ResultSet query(AbsRdbPredicates predicates, String[] columns)	查询数据

续表

方　　法	描　　述
ResultSet querySql(String sql, String[] sqlArgs)	执行原生的用于查询操作的 SQL 语句
int delete(AbsRdbPredicates predicates)	删除数据
void executeSql(String sqlString)	执行原生的 SQL 语句

AbsRdbPredicates 是关系数据库提供的用于设置数据库操作条件的谓词，其中包括两个实现子类 RdbPredicates 和 RawRdbPredicates。

- RdbPredicates：开发者无须编写复杂的 SQL 语句，仅通过调用该类中条件相关的方法，如 equalTo()、notEqualTo()、groupBy()、orderByAsc()、beginsWith()等，就可自动完成 SQL 语句拼接，方便用户聚焦业务操作。
- RawRdbPredicates：可满足复杂 SQL 语句的场景，支持开发者自己设置 where 条件子句和 whereArgs 参数，不支持 equalTo 等条件接口的使用。

RdbPredicates 类定义的常用操作方法如表 5-4 所示。RawRdbPredicates 类定义的常用操作方法如表 5-5 所示。

表 5-4　RdbPredicates 类定义的常用操作方法

方　　法	描　　述
RdbPredicates equalTo(String field, String value)	设置谓词条件，满足 field 字段与 value 值相等
RdbPredicates notEqualTo(String field, String value)	设置谓词条件，满足 field 字段与 value 值不相等
RdbPredicates beginsWith(String field, String value)	设置谓词条件，满足 field 字段以 value 值开头
RdbPredicates between(String field, int low, int high)	设置谓词条件，满足 field 字段在最小值 low 和最大值 high 之间
RdbPredicates notBetween(String field, int low, int high)	设置谓词条件，满足 field 字段不在最小值 low 和最大值 high 之间
RdbPredicates orderByAsc(String field)	设置谓词条件，根据 field 字段升序排列
RdbPredicates orderByDesc(String field)	设置谓词条件，根据 field 字段降序排列
RdbPredicates greaterThan(String field, String value)	设置谓词条件，满足 field 字段比 value 值大
RdbPredicates lessThan(String field, String value)	设置谓词条件，满足 field 字段比 value 值小

表 5-5　RawRdbPredicates 类定义的常用操作方法

方　　法	描　　述
void setWhereClause(String whereClause)	设置 where 条件子句
void setWhereArgs(List<String> whereArgs)	设置 whereArgs 参数，该值表示 where 子句中占位符的值

ResultSet 是关系数据库提供查询返回的结果集，它指向查询结果中的一行数据，供

用户对查询结果进行遍历和访问。ResultSet 类定义的常用操作方法如表 5-6 所示。

表 5-6　ResultSet 类定义的常用操作方法

方　法	描　述
boolean goTo(int offset)	从结果集当前位置移动指定偏移量
boolean goToRow(int position)	将结果集移动到指定位置
boolean goToFirstRow()	将结果集移动到第一行
boolean goToLastRow()	将结果集移动到最后一行
boolean goToNextRow()	将结果集向后移动一行
boolean goToPreviousRow()	将结果集向前移动一行
boolean isStarted()	判断结果集是否被移动过
boolean isEnded()	判断结果集当前位置是否在最后一行之后
boolean isAtFirstRow()	判断结果集当前位置是否在第一行
boolean isAtLastRow()	判断结果集当前位置是否在最后一行
int getRowCount()	获取当前结果集中的记录条数
int getColumnCount()	获取结果集中的列数
String getString(int columnIndex)	获取当前行指定列的值，以 String 类型返回
int getInt(int i)	获取当前行指定列的值，以 int 类型返回
void close()	关闭结果集
int getColumnIndexForName(String s)	获取结果集中列表名为 s 的列号
String getColumnNameForIndex(int i)	获取结果集中列号为 i 的列表名

5.1.4　关系数据库开发步骤

下面讲解关系数据库的创建，以及数据的增加、删除、修改、查询等基本步骤。

(1) 首先创建数据库，配置数据库的相关信息，初始化数据库的表结构和相关数据，示例代码如程序清单 5-1 所示。

程序清单 5-1：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\slice\MainAbilitySlice.java

```
1  StoreConfig storeConfig = StoreConfig.newDefaultConfig("RdbStoreTest.db");
2  DatabaseHelper databaseHelper= new DatabaseHelper(context);
3  private static RdbOpenCallback rdbOpenCallback = new RdbOpenCallback() {
4      @Override
5      public void onCreate(RdbStore rdbStore) {
6          //创建表
```

```
7           rdbStore.executeSql("CREATE TABLE IF NOT EXISTS testdb (userId
    INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT NOT NULL, userAge
    INTEGER)");
8       }
9       @Override
10      public void onUpgrade(RdbStore rdbStore, int oldVersion, int
    newVersion) {
11          //升级数据库操作
12      }
13  };
14  RdbStore rdbStore = databaseHelper.getRdbStore(storeConfig, 1,
    rdbOpenCallback, null);
```

(2) 插入数据操作,以 ValuesBucket 形式构造要插入的数据,示例代码如程序清单 5-2 所示。

程序清单 5-2：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\slice\MainAbilitySlice.java

```
1   ValuesBucket values = new ValuesBucket();
2   values.putInteger("userId", 1);
3   values.putString("userName", "Jerry");
4   values.putInteger("userAge", 18);
5   long id = rdbStore.insert("User", values);
6   //使用 SQL 语句插入数据
7   rdbStore.executeSql("insert into User (userId,userName,userAge) values
    (2,'Tom',20)");
```

(3) 删除数据操作,详细代码如程序清单 5-3 所示。

程序清单 5-3：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\slice\MainAbilitySlice.java

```
1   RdbPredicates rdbPredicates = new RdbPredicates("User").equalTo
    ("userName","Jerry");
2   int i = rdbStore.delete(rdbPredicates);
```

(4) 查询数据操作,详细代码如程序清单 5-4 所示。

程序清单 5-4：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\slice\MainAbilitySlice.java

```
1   //按条件查询
2   String[] columns = new String[] {"userId", "userName", "userAge"};
```

```
RdbPredicates rdbPredicates = new RdbPredicates("User").equalTo
("userAge", 18).orderByAsc("userId");
ResultSet resultSet = store.query(rdbPredicates, columns);
resultSet.goToNextRow();
//查询所有数据
String[] columns = new String[]{"userId","userName","userAge"};
RdbPredicates rdbPredicates = new RdbPredicates("User");//构建查询谓词
ResultSet resultSet = rdbCreateDb().query(rdbPredicates,columns);
while (resultSet.goToNextRow()){
    int userId = resultSet.getInt(resultSet.getColumnIndexForName
("userId"));
    String userName = resultSet.getString(resultSet.
getColumnIndexForName("userName"));
    int userAge = resultSet.getInt(resultSet.getColumnIndexForName
("userAge"));
}
```

(5) 更新数据操作,详细代码如程序清单 5-5 所示。

程序清单 5-5：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\slice\MainAbilitySlice.java

```
RdbPredicates rdbPredicates = new RdbPredicates("User").equalTo
("userName","Tom");
ValuesBucket values = new ValuesBucket();
values.putString("userName","Lee");
//更新数据
rdbStore.update(values,rdbPredicates);
```

5.1.5 基于 Data Ability 的关系数据库操作案例

前面讲解了操作 HarmonyOS 关系数据库的相关类以及关系数据库的基本开发步骤。本节将实现基于 Data Ability 创建数据库服务,对外提供访问数据库的服务端接口(PersonInfoDataAbility 类),并在 MainAbilitySlice 中通过 DataAbilityHelper 与提供方的 Data Ability 进行通信,最后通过日志查看数据库操作结果。

1. 创建一个 Data Ability

在新建的名为 RelationalDatabaseTest 的工程的主目录下添加一个名为 dataability 的包,在 dataability 下添加一个名为 PersonInfoDataAbility 的 Empty Data Ability,用于存储人员信息数据库并提供接口。HUAWEI DevEco Studio 将会自动生成数据库的 CRUD(增加、删除、修改、查询)方法,详细代码如程序清单 5-6 所示。

程序清单 5-6：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\dataability\PersonInfoDataAbility.java

```
1  public class PersonInfoDataAbility extends Ability {
2      private static final HiLogLabel LABEL_LOG = new HiLogLabel(3,
   0xD001100, "Demo");
3
4      @Override
5      public void onStart(Intent intent) {
6          super.onStart(intent);
7          HiLog.info(LABEL_LOG, "PersonInfoDataAbility onStart");
8      }
9
10     @Override
11     public ResultSet query(Uri uri, String[] columns,
   DataAbilityPredicates predicates) {
12         return null;
13     }
14
15     @Override
16     public int insert(Uri uri, ValuesBucket value) {
17         HiLog.info(LABEL_LOG, "PersonInfoDataAbility insert");
18         return 999;
19     }
20
21     @Override
22     public int delete(Uri uri, DataAbilityPredicates predicates) {
23         return 0;
24     }
25
26     @Override
27     public int update(Uri uri, ValuesBucket value, DataAbilityPredicates
   predicates) {
28         return 0;
29     }
30 }
```

同时，HUAWEI DevEco Studio 将自动在工程配置文件 config.json 中添加如程序清单 5-7 所示的配置。

程序清单 5-7：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\config.json

```
1  {
2    "permissions": [
```

```
    "com.example.relationaldatabasetest.DataAbilityShellProvider.
PROVIDER"
  ],
  "name": "com.example.relationaldatabasetest.dataability.
PersonInfoDataAbility",
  "icon": "$media:icon",
  "description": "empty provider",
  "type": "data",
  "uri": "dataability://com.example.relationaldatabasetest.dataability.
PersonInfoDataAbility"
}
```

2. 定义 Data Ability 数据库相关常量

为了方便后续操作，在 PersonInfoDataAbility 类中定义数据库的相关常量，包括数据库的库名、表名、表字段名、数据库版本号等，详细代码如程序清单 5-8 所示。

程序清单 5-8：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\dataability\PersonInfoDataAbility.java

```
public class PersonInfoDataAbility extends Ability {
...
    private static final String DB_NAME = "personinfodataability.db";
                                                          //数据库名
    private static final String DB_TAB_NAME = "personinfo";  //表名
    private static final String DB_COLUMN_PERSON_ID = "id";
    private static final String DB_COLUMN_NAME = "name";
    private static final String DB_COLUMN_GENDER = "gender";
    private static final String DB_COLUMN_AGE = "age";
    private static final int DB_VERSION = 1;
...
}
```

3. 创建关系数据库

在 PersonInfoDataAbility 类中定义 RdbStore 变量，并通过 RdbOpenCallback 创建数据库 personinfodataability.db 及表 personinfo，详细代码如程序清单 5-9 所示。

程序清单 5-9：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\dataability\PersonInfoDataAbility.java

```
private StoreConfig config = StoreConfig.newDefaultConfig(DB_NAME);
private RdbStore rdbStore;
private RdbOpenCallback rdbOpenCallback = new RdbOpenCallback() {
    @Override
    public void onCreate(RdbStore store) {
```

```
6           store.executeSql("create table if not exists "
7                   + DB_TAB_NAME + " ("
8                   + DB_COLUMN_PERSON_ID + " integer primary key, "
9                   + DB_COLUMN_NAME + " text not null, "
10                  + DB_COLUMN_GENDER + " text not null, "
11                  + DB_COLUMN_AGE + " integer)");
12      }
13
14      @Override
15      public void onUpgrade(RdbStore store, int oldVersion, int
    newVersion) {
16      }
17  };
```

4. 初始化数据库连接

在程序应用启动时，系统会调用 Data Ability 中的 onStart()方法创建 Data 实例。在此方法中，需要创建数据库连接，并获取连接对象，以便后续对数据库进行操作，详细代码如程序清单 5-10 所示。

程序清单 5-10：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\dataability\PersonInfoDataAbility.java

```
1   @Override
2   public void onStart(Intent intent) {
3       super.onStart(intent);
4       HiLog.info(LABEL_LOG, "PersonInfoDataAbility onStart");
5       DatabaseHelper databaseHelper = new DatabaseHelper(this);
6       rdbStore = databaseHelper.getRdbStore(config, DB_VERSION,
    rdbOpenCallback, null);
7   }
```

5. 重写数据库操作方法

创建 Data Ability 时，HUAWEI DevEco Studio 自动生成数据库增加、删除、修改、查询的空方法，开发者可以按照自身需要重写相关方法。

(1) query()：查询数据方法。

该方法接收 3 个参数，分别为 uri(查询的目标路径)、columns(查询的列名)、predicates(查询条件)。其中查询条件由 DataAbilityPredicates 类构建，详细代码如程序清单 5-11 所示。

程序清单 5-11：hmos\ch05\01\RelationalDatabaseTest\entry\src\main\java\com\example\relationaldatabasetest\dataability\PersonInfoDataAbility.java

```
1   @Override
```