Harmony OS
入门系列课程 <快速上手>

快速掌握鸿蒙系统应用开发基础操作技巧

第 7 讲：鸿蒙OS数据库基本操作

章节目录

- 名词解析
- 概述及运作机制
- 关系型数据库相关接口说明
- 关系映射数据库相关接口说明
- 代码解析
- 小结及巩固练习

课程目标

- 掌握鸿蒙OS关系型数据库相关知识及开发流程；

- 掌握鸿蒙OS关系映射数据库相关知识及开发流程；

名词概念

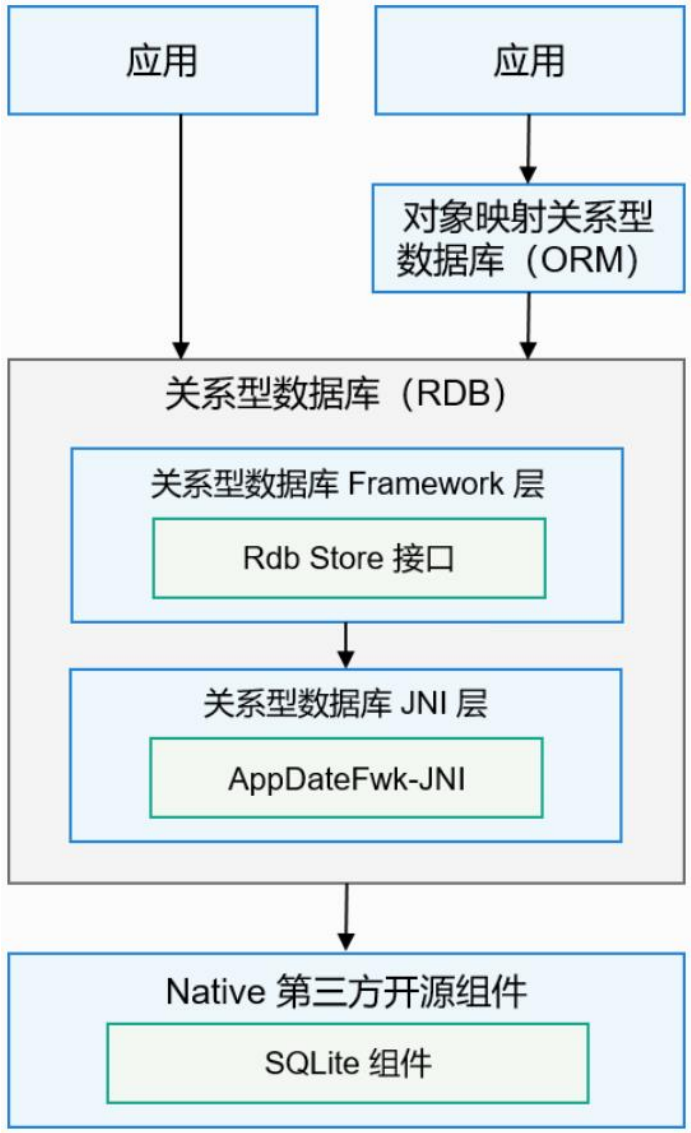
- **SQLite数据库**：一款轻型的数据库，是遵守ACID的关系型数据库管理系统。它是一个开源的项目。
- **关系型数据库**：创建在关系模型基础上的数据库，以行和列的形式存储数据。
- **ORM**：Object Relational Mapping，对象关系映射。
- **对象关系映射数据库**：通过将实例对象映射到关系上，实现使用操作实例对象的语法，来操作关系型数据库。它是在SQLite数据库的基础上提供的一个抽象层。
- **谓词**：数据库中用来代表数据实体的性质、特征或者数据实体之间关系的词项，主要用来定义数据库的操作条件。对象关系映射数据库将SQLite数据库中的谓词封装成了接口方法供开发者调用。开发者通过对象数据操作接口，可以访问到应用持久化的关系型数据。
- **结果集**：用户查询之后的结果集合，可以对数据进行访问。结果集提供了灵活的数据访问方式，可以更方便的拿到用户想要的数据。

概述

- HarmonyOS关系型数据库基于SQLite组件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、删、改、查接口，也可以直接运行用户输入的SQL语句来满足复杂的场景需要。
- HarmonyOS对象关系映射数据库是一款基于SQLite的数据库框架，屏蔽了底层SQLite数据库的SQL操作，针对实体和关系提供了增删改查等一系列的面向对象接口。应用开发者不必再去编写复杂的SQL语句，以操作对象的形式来操作数据库，提升效率的同时也能聚焦于业务开发。
- 对象关系映射数据库目前可以支持数据库和表的创建，对象数据的增删改查、对象数据变化回调、数据库升降级和备份等功能。其三个主要组件如下：

组件名	说明
数据库	被开发者用@Database注解，且继承了OrmDatabase的类，对应关系型数据库。
实体对象	被开发者用@Entity注解，且继承了OrmObject的类，对应关系型数据库中的表。
对象数据操作接口	包括数据库操作的入口OrmContext类和谓词接口（OrmPredicate）等。

关系型数据库运作机制


应用
应用
对象映射关系型数据库（ORM）
关系型数据库（RDB）
关系型数据库 Framework 层
Rdb Store 接口
关系型数据库 JNI 层
AppDateFwk-JNI
Native 第三方开源组件
SQLite 组件

关系型数据库--接口说明

关系型数据库提供了数据库创建方式，以及对应的删除接口，涉及的API如下所示：

类名	接口名	描述
StoreConfig.Builder	public builder()	对数据库进行配置，包括设置数据库名、存储模式、日志模式、同步模式，是否为只读，及对数据库加密。
	Builder setEncryptKey(byte[] encryptKey)	为数据库配置类设置数据库加密密钥，创建或打开数据库时传入包含数据库加密密钥的配置类，即可创建或打开加密数据库。
RdbOpenCallback	public abstract void onCreate(RdbStore store)	数据库创建时被回调，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 void onUpgrade(RdbStore store, int currentVersion, int targetVersion)	数据库升级时被回调。
DatabaseHelper	public RdbStore getRdbStore(StoreConfig config, int version, RdbOpenCallback openCallback, ResultSetHook resultSetHook)	根据配置创建或打开数据库。
	public boolean deleteRdbStore(String name)	删除指定的数据库。

关系型数据库--增删改查接口

关系型数据库提供本地数据增删改查操作的能力，相关API如下所示：

类名	接口名	描述
RdbStore	long insert(String table, ValuesBucket initialValues)	插入数据的接口，通过ValuesBucket输入要存储的数据，通过返回值判断是否插入成功，插入成功时返回最新插入数据所在的行号，失败则返回-1。
	int update(ValuesBucket values, AbsRdbPredicates predicates)	更新接口，通过ValuesBucket传入要更新的数据，并通过AbsRdbPredicates指定更新条件。该接口的返回值表示更新操作影响的行数。如果更新失败，则返回0。 AbsRdbPredicates的实现类有两个：RdbPredicates和RawRdbPredidates。
	int delete(AbsRdbPredicates predicates)	删除接口，通过AbsRdbPredicates指定删除条件。该接口的返回值表示删除的数据行数，可根据此值判断是否删除成功。如果删除失败，则返回0。
	ResultSet query(AbsRdbPredicates predicates, String[] columns)	直接调用查询接口。使用该接口，会将包含查询条件的谓词自动拼接成完整的SQL语句进行查询操作，无需用户传入原生的SQL。
	ResultSet querySql(String sql, String[] sqlArgs)	执行原生的用于查询的SQL语句。 sqlArgs表示语句中占位符参数的值，如语句中无占位符，可设为null。

关系型数据库--数据库谓词

- **AbsRdbPredicates**：关系型数据库中用于设置数据库操作条件的谓词，包括两个实现子类：
  - **RdbPredicates**：开发者无需编写复杂的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。

关系型数据库--数据库谓词API

类名	接口名	描述
RdbPredicates	RdbPredicates equalTo(String field, String value)	设置谓词条件，满足filed字段与value值相等。
	RdbPredicates notEqualTo(String field, String value)	设置谓词条件，满足filed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 orderByAsc(String field)	设置谓词条件，根据field字段升序排列。
RawRdbPredicates	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。

关系型数据库--查询结果集API

类名	接口名	描述
ResultSet	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 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)	获取当前行指定索列的值，以String类型返回。
	byte[] getBlob(int columnIndex)	获取当前行指定列的值，以字节数组形式返回。
	double getDouble(int columnIndex)	获取当前行指定列的值，以double型返回。

关系型数据库--数据库事务API

关系型数据库提供事务机制，来保证用户操作的原子性。对单条数据进行数据库操作时，无需开启事务；插入大量数据时，开启事务可以保证数据的准确性。如果中途操作出现失败，会执行回滚操作。

关系型数据库提供了事务和结果集观察者能力，当对应的事件被触发时，观察者会收到通知。

类名	接口名	描述
RdbStore	beginTransaction()	开启事务。
	markAsCommit()	设置事务的标记为成功。
	endTransaction()	结束事务。
	beginTransactionWithObserver(TransactionObserver transactionObserver)	开启事务，并观察事务的启动、提交和回滚。
ResultSet	void registerObserver(DataObserver observer)	注册结果集的观察者。
	void unregisterObserver(DataObserver observer)	注销结果集的观察者。

关系型数据库--数据库备份和恢复

用户可以将当前数据库的数据进行保存进行备份，还可以在需要的时候进行数据恢复。

类名	接口名	描述
RdbStore	boolean restore(String srcName)	数据库恢复接口，从指定的非加密数据库文件中恢复数据。
	boolean restore(String srcName, byte[] srcEncryptKey, byte[] destEncryptKey)	数据库恢复接口，从指定的数据库文件（加密和非加密均可）中恢复数据。
	boolean backup(String destName)	数据库备份接口，备份出的数据库文件是非加密的。
	boolean backup(String destName, byte[] destEncryptKey)	数据库备份接口，此方法经常用在备份出加密数据库场景。

关系型数据库--开发步骤解析

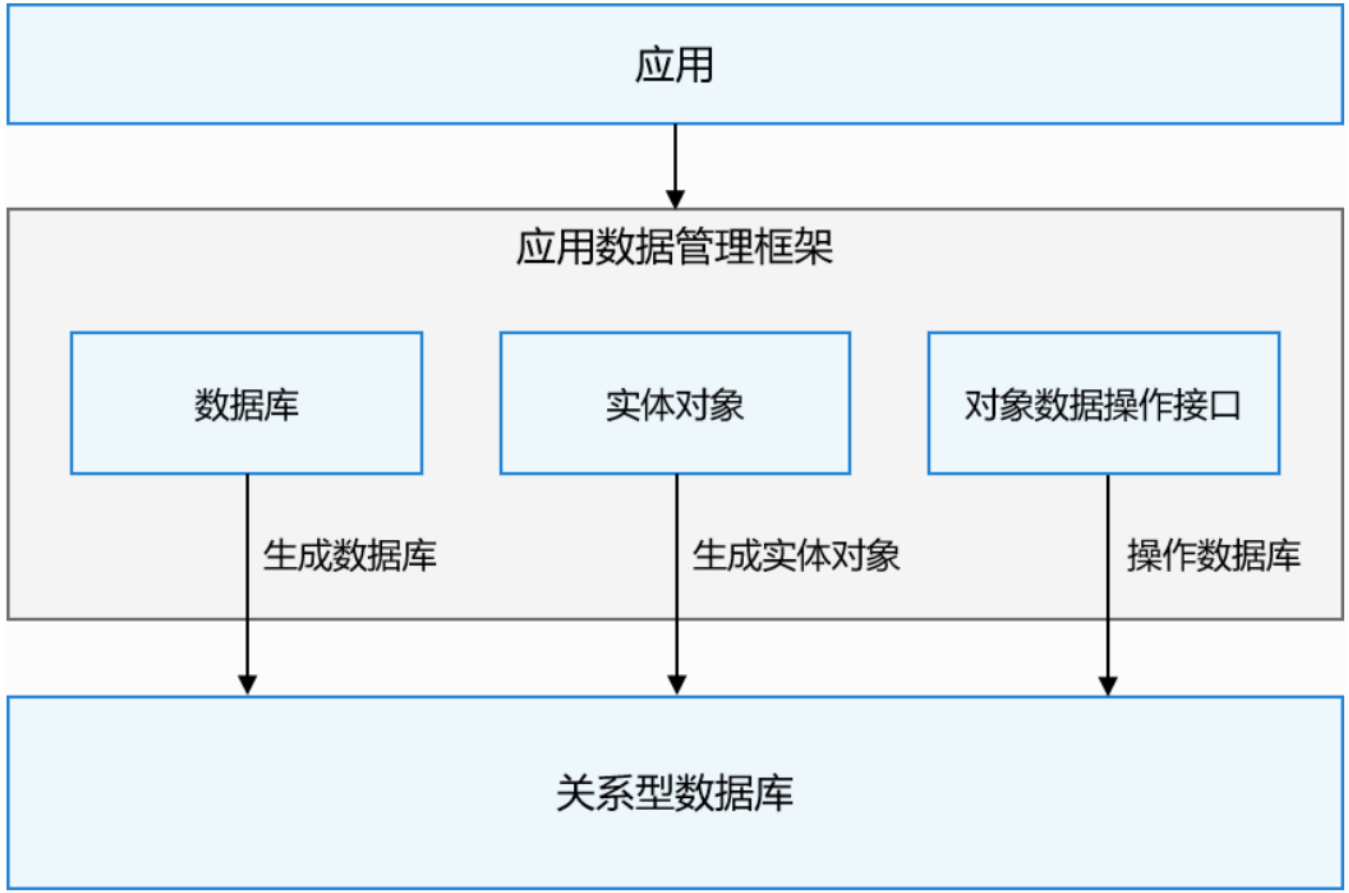
- 1. 创建数据库：
  ① 配置数据库相关信息，包括数据库的名称、存储模式、是否为只读模式等。
  ② 初始化数据库表结构和相关数据。
  ③ 创建数据库。

- 2. 插入数据：
  ① 构造要插入的数据，以ValuesBucket形式存储。
  ② 调用关系型数据库提供的插入接口插入数据。

- 3. 查询数据：
  ① 构造用于查询的谓词对象，设置查询条件。
  ② 指定查询返回的数据列。
  ③ 调用查询接口查询数据。
  ④ 调用结果集接口，遍历返回结果。

对象关系映射数据库运作机制

对象关系映射数据库适用于开发者使用的数据可以分解为一个或多个对象，且需要对数据进行增删改查等操作，但是不希望编写过于复杂的SQL语句的场景。



常用实体对象属性支持的类型

建立实体对象类时，对象属性的类型可以在下表的类型中选择。不支持使用自定义类型。

类型名称	描述	初始值
Short/short	封装短整型/短整型	null/0
Integer/int	封装整型/整型	null/0
Long/long	封装长整型/长整型	null/0L
Double/double	封装双精度浮点型/双精度浮点型	null/0
Float/float	封装单精度浮点型/单精度浮点型	null/0
Boolean/boolean	封装布尔型/布尔型	null/0
Byte/byte	封装字节型/字节型	null/0
Character/char	封装字符型/字符型	null/' '

类型名称	描述	初始值
String	字符串型	null
Date	日期类	null
Time	时间类	null
Timestamp	时间戳类	null
Calendar	日历类	null
Blob	二进制大对象	null
Clob	字符大对象	null

对象关系映射数据库--数据库和表的创建

注解对照表如下：

接口名称	说明
@Database	被@Database注解且继承了OrmDatabase的类对应数据库类。
@Entity	被@Entity注解且继承了OrmObject的类对应数据表类。
@Column	被@Column注解的变量对应数据表的字段。
@PrimaryKey	被@PrimaryKey注解的变量对应数据表的主键。
@ForeignKey	被@ForeignKey注解的变量对应数据表的外键。
@Index	被@Index注解的内容对应数据表索引的属性。

对象关系映射数据库--数据库的加密

对象关系映射数据库提供数据库加密的能力，创建数据库时传入指定密钥、创建加密数据库，后续打开加密数据库时，需要传入正确密钥。

类名	接口名称	说明
OrmConfig.Builder	Builder setEncryptKey(byte[] encryptKey)	为数据库配置类设置数据库加密密钥，创建或打开数据库时传入包含数据库加密密钥的配置类，即可创建或打开加密数据库。

对象关系映射数据库--对象数据的增删改查

对象数据操作接口如下：

类名	接口名称	说明
OrmContext	<T extends OrmObject> boolean insert(T object)	添加
	<T extends OrmObject> boolean update(T object)	更新
	<T extends OrmObject> List<T> query(OrmPredicates predicates)	查询
	<T extends OrmObject> boolean delete(T object)	删除
	<T extends OrmObject> OrmPredicates where(Class<T> clz)	设置谓词

对象关系映射数据库--对象数据的变化观察者设置

通过使用对象数据操作接口，开发者可以在某些数据上设置观察者，接收数据变化的通知。

类名	接口名称	说明
OrmContext	void registerStoreObserver(String alias, OrmObjectObserver observer)	注册数据库变化回调
	void registerContextObserver(OrmContext watchedContext, OrmObjectObserver observer)	注册上下文变化回调
	void registerEntityObserver(String entityName, OrmObjectObserver observer)	注册数据库实体变化回调
	void registerObjectObserver(OrmObject ormObject, OrmObjectObserver observer)	注册对象变化回调

对象关系映射数据库--数据库的升降级

通过调用数据库升降级接口，开发者可以将数据库切换到不同的版本。

类名	接口名称	说明
OrmMigration	public void onMigrate(int beginVersion, int endVersion)	数据库版本升降级接口

对象关系映射数据库--数据库的备份恢复

将当前数据库的数据进行备份，在必要的时候进行数据恢复。

类名	接口名称	说明
OrmContext	boolean backup(String destPath)	数据库备份接口
	boolean restore(String srcPath)	数据库恢复备份接口

对象关系映射数据库--开发步骤解析

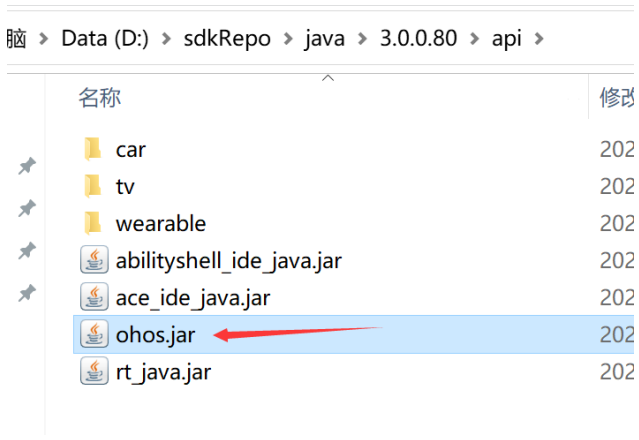
- 一、配置“build.gradle”文件
  - 如果使用注解处理器的模块为“com.huawei.ohos.hap”模块，则需要在模块的“build.gradle”文件的“ohos”节点中添加以下配置：

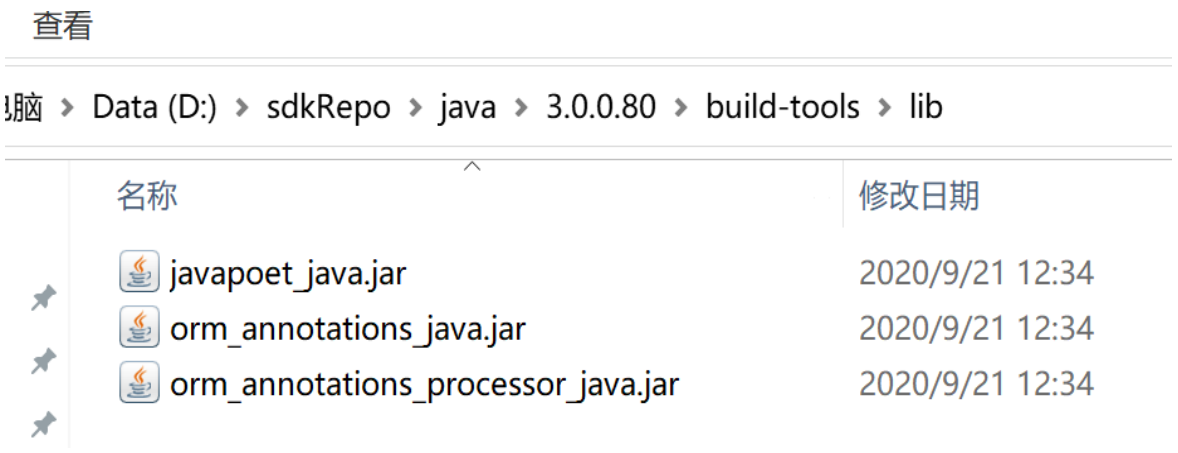
```
compileOptions{
    annotationEnabled true
}
```

  - 如果使用注解处理器的模块为“com.huawei.ohos.library”模块，则需要在模块的“build.gradle”文件的“dependencies”节点中配置注解处理器。查看“orm_annotations_java.jar”、“orm_annotations_processor_java.jar”、“javapoet_java.jar”这3个jar包在HUAWEI SDK中的对应目录，并将目录的这三个jar包导进来。

  - 如果使用注解处理器的模块为“java-library”模块，则需要在模块的“build.gradle”文件的“dependencies”节点中配置注解处理器，并导入“ohos.jar”。

Jar包及配置说明






```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    testCompile'junit:junit:4.12'

    compile files(
            "D:\\sdkRepo\\java\\3.0.0.80\\api\\ohos.jar",
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\orm_annotations_java.jar",
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\orm_annotations_processor_java.jar",
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\javapoet_java.jar")
    annotationProcessor files(
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\orm_annotations_java.jar",
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\orm_annotations_processor_java.jar",
            "D:\\sdkRepo\\java\\3.0.0.80\\build-tools\\lib\\javapoet_java.jar")
}
```

对象关系映射数据库--开发步骤解析

- 二、构造数据库
  - 即创建数据库类并配置对应的属性；
  - 添加@Database注解并继承OrmDatabase；
  - 数据库类的getVersion方法和getHelper方法不需要实现，直接将数据库类设为抽象类即可；
  - 例如，定义了一个数据库类BookStore.java，数据库包含了“User”，"Book"，"AllDataType"三个表，版本号为 “1”。

```
@Database(entities = {User.class, Book.class, AllDataType.class}, version = 1)
public abstract class BookStore extends OrmDatabase {
}
```

对象关系映射数据库--开发步骤解析

三、构造数据表

- 创建数据库实体类并配置对应的属性（如对应表的主键，外键等）；
- 数据表必须与其所在的数据库在同一个模块中；
- 例如，定义了一个实体类User.java，对应数据库内的表名为“user”；indices 为“firstName”和“lastName”两个字段建立了复合索引“name_index”，并且索引值是唯一的；“ignoreColumns”表示该字段不需要添加到“user”表的属性中。

```
@Entity(tableName = "user", ignoredColumns = {"ignoreColumn1", "ignoreColumn2"},
    indices = {@Index(value = {"firstName", "lastName"}, name = "name_index", unique = true)})
public class User extends OrmObject {
    // 此处将userId设为了自增的主键。注意只有在数据类型为包装类型时，自增主键才能生效。
    @PrimaryKey(autoGenerate = true)
    private Integer userId;
    private String firstName;
    private String lastName;
    private int age;
    private double balance;
    private int ignoreColumn1;
    private int ignoreColumn2;

    // 开发者自行添加字段的getter和setter 方法。
}
```

对象关系映射数据库--开发步骤解析

- 四、使用对象数据操作接口OrmContext创建数据库

  - 例如，通过对象数据操作接口OrmContext，创建一个别名为“BookStore”，数据库文件名为“BookStore.db”的数据库。如果数据库已经存在，执行以下代码不会重复创建。通过context.getDatabaseDir()可以获取创建的数据库文件所在的目录。

```
DatabaseHelper helper = new DatabaseHelper(context);
OrmContext context = helper.getOrmContext("BookStore", "BookStore.db", BookStore.class);
```

  - 注意：context入参类型为ohos.app.Context，注意不要使用slice.getContext()来获取context，请直接传入slice，否则会出现找不到类的报错。

对象关系映射数据库--开发步骤解析

- 五、数据操作（增删改查）
  - 使用对象数据操作接口OrmContext对数据库进行增删改查。
  - 增加数据示例代码如下，可直接传入OrmObject对象的增加接口，但只有在**flush()接口**被调用后才会持久化到数据库中：

```
User user = new User();
user.setFirstName("Zhang");
user.setLastName("San");
user.setAge(29);
user.setBalance(100.51);
boolean isSuccessed = context.insert(user);
isSuccessed = context.flush();
```

对象关系映射数据库--开发步骤解析

- 五、更新或删除数据，分为两种情况：
  - 通过直接传入OrmObject对象的接口来更新数据，需要先从表中查到需要更新的User对象列表，然后修改对象的值，再调用update接口传入被更新的User对象。最后调用flush接口持久化到数据库中。删除数据与更新数据的方法类似，只是不需要更新对象的值。

```
// 更新数据
OrmPredicates predicates = context.where(User.class);
predicates.equalTo("age",29);
List<User> users = context.query(predicates);
User user = users.get(0);
user.setFirstName("Li");
context.update(user);
context.flush();

// 删除数据
OrmPredicates predicates = context.where(User.class);
predicates.equalTo("age",29);
List<User> users = context.query(predicates);
User user = users.get(0);
context.delete(user);
context.flush();
```

对象关系映射数据库--开发步骤解析

- 五、更新或删除数据，分为两种情况：
  - 通过传入谓词的接口来更新和删除数据，方法与OrmObject对象的接口类似，只是**无需flush**就可以持久化到数据库中。

```
ValuesBucket valuesBucket = new ValuesBucket();
valuesBucket.putInteger("age", 31);
valuesBucket.putString("firstName", "ZhangU");
valuesBucket.putString("lastName", "SanU");
valuesBucket.putDouble("balance", 300.51);
OrmPredicates update = context.where(User.class).equalTo("userId", 1);
context.update(update, valuesBucket);
```

对象关系映射数据库--开发步骤解析

- 六、查询数据
  - 在数据库的“user”表中查询lastName为“San”的User对象列表，示例如下：

```
OrmPredicates query = context.where(User.class).equalTo("lastName", "San");
List<User> users = context.query(query);
```

对象关系映射数据库--开发步骤解析

- 七、备份数据库
  - 原数据库名为“OrmBackUp.db”，备份数据库名为“OrmBackup001.db”，示例如下：

```
OrmContext context = helper.getObjectContext("OrmBackup", "OrmBackup.db", BookStore.class);
context.backup("OrmBackup001.db");
context.close();
```

对象关系映射数据库--开发步骤解析

- 八、删除数据库
  - 删除“OrmBackup.db”，示例如下：

```
helper.deleteRdbStore("OrmBackup.db");
```

内容小结

- 鸿蒙OS关系型数据库的接口及其使用步骤
- 鸿蒙OS对象关系映射数据库的接口及其使用步骤

练习巩固

问题1：以下哪项不属于对象关系映射数据库创建数据表的属性？（ ）

A. tableName

B. primaryKey

C. foreignKeys

D. indices

问题2：谓词AbsRdbPredicates的实现子类RdbPredicates是可满足复杂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。

A. 正确

B. 错误

思考挑战

一、创建一个动物园管理系统，管理员可以向动物园中新增、修改、删除动物信息，可以查看所有动物信息或查看某一动物的信息。请通过鸿蒙对象关系数据库实现。

二、思考如何实现一个数据库操作通用的工具类?

THANKS

更多学习视频，关注宅客学院……