caching with Sqflite in flutter

هي احد package الخاصه ب flutter للقيام بتغزين البيانات في ذاكره الهاتف و لكن تتميز ب انها تعتمد علي تعليمات sql للقيام بالمهام المراده سوء select او delete او update او insert

*import* 'package:sqflite/sqflite.dart'**;**

*import* 'package:path/path.dart'**;**

class SqlDb {

  SqlDb.\_()**;**

*static* *final* SqlDb instance **=** SqlDb.\_()**;**

*static* Database**?** \_db**;**

  Future<Database?> get db **async** {

**if** (\_db **==** null) {

      \_db **=** **await** intialDb()**;**

**return** \_db**;**

    } **else** {

**return** \_db**;**

    }

  }

  intialDb() **async** {

    String databasepath **=** **await** getDatabasesPath()**;**

    String path **=** join(databasepath, 'todo.db')**;**

    Database mydb **=** **await** openDatabase(path,

        onCreate**:** \_onCreate, version**:** 1, onUpgrade**:** \_onUpgrade)**;**

**return** mydb**;**

  }

  \_onUpgrade(Database db, int oldversion, int newversion) **async** {

    print('onupgrade======================================')**;**

  }

  \_onCreate(Database db, int version) **async** {

**await** db.execute('''

  CREATE TABLE "task" (

    "t\_id" INTEGER  NOT NULL PRIMARY KEY  AUTOINCREMENT,

    "t\_taskt" TEXT NOT NULL,

    "t\_note" TEXT NOT NULL DEFAULT 'empty',

    "t\_date" DATE NOT NULL,

    "t\_time" DATETIME NOT NULL,

    "t\_done" TINYINT NOT NULL DEFAULT '0',

    "t\_categ" INTEGER NOT NULL

  )

 ''')**;**

**await** db.execute('''

  CREATE TABLE "category" (

    "c\_id" INTEGER  NOT NULL PRIMARY KEY  AUTOINCREMENT,

    "c\_categ" TEXT NOT NULL DEFAULT 'empty',

    "c\_hide" TINYINT NOT NULL DEFAULT '0'

  )

  ''')**;**

**await** db.execute('''

  CREATE TABLE "startask" (

    "s\_id" INTEGER  NOT NULL PRIMARY KEY  AUTOINCREMENT,

    "s\_idtask" INTEGER NOT NULL,

    FOREIGN KEY (s\_idtask) REFERENCES task(t\_id)

  )

  ''')**;**

    print('oncreate======================================')**;**

  }

  readData(String table, String**?** where) **async** {

    Database**?** mydb **=** **await** db**;**

    List<Map> response **=** **await** mydb**!**.query(table, where**:** where)**;**

**return** response**;**

  }

  rawreadData(String sql) **async** {

    Database**?** mydb **=** **await** db**;**

    List<Map> response **=** **await** mydb**!**.rawQuery(sql)**;**

**return** response**;**

  }

  insertData(String table, Map<String, Object?> values) **async** {

    Database**?** mydb **=** **await** db**;**

    int response **=** **await** mydb**!**.insert(table, values)**;**

**return** response**;**

  }

  updateData(String table, Map<String, Object?> values, String**?** where) **async** {

    Database**?** mydb **=** **await** db**;**

    int response **=** **await** mydb**!**.update(table, values, where**:** where)**;**

**return** response**;**

  }

  deleteData(String table, String**?** where) **async** {

    Database**?** mydb **=** **await** db**;**

    int response **=** **await** mydb**!**.delete(table, where**:** where)**;**

**return** response**;**

  }

  deletedatabases() **async** {

    String databasepath **=** **await** getDatabasesPath()**;**

    String path **=** join(databasepath, 'todo.db')**;**

**return** deleteDatabase(path)**;**

  }

}