Sqlite - By Dorayo

简介

SQLite是个轻量级的嵌入式关系数据库。它不作为一个独立的进程运行，而是通过动态或者静态库的方式链接到应用程序中。它生成的数据库文件是一个普通的磁盘文件，可以放在任何目录下。SQLite本身是C语言开发的，开放源码，并且跨平台支持，并且被大部分驻留编程语言支持。详细内容-> sqlite.org

SQLite数据库之SQL语言

数据定义语言(DDL)

- CREATE
- ALTER TABLE
- DROP

数据操作语言(DML)

- INSERT
- UPDATE
- DELETE

数据查询语言(DQL)

- SELECT

SQLite中的数据类型

- NULL 空值
- INTEGER 有符号整数
- REAL 浮点数
- TEXT 文本字符串
- BLOB 二进制数据，如图片、声音等

SQLite命令行工具安装

- Windows版本：下载
- Unix/Linux：终端中自带sqlite，无需下载安装

SQLite命令行工具

数据库和表的元命令

1. 创建数据库 `sqlite3 test.db`
2. 打开已经存在的数据库 `sqlite3 test.db`
3. 导入数据 `.read test.sql`
4. 列出所有数据表：`.tables`
5. 显示数据库结构：`.schema`
6. 显示表的结构：`.schema Students`
7. 导出某个表的数据：`.dump Students`
8. 设置导出文件流：`.output stdout`

数据显示相关元命令

1. 设置分隔符：`.separator :`
2. 设置显示模式：`.mode column`
3. 显示标题栏：`.headers on`
4. 设置每列的显示宽度：`.width 8, 10, 15`
5. 设置NULL值得显示样式：`.nullvalue NaN`
6. 列出当前显示格式的设置情况：`.show`
7. 配置文件：`~/.sqliterc`

SQLite数据库操作

DDL

创建表 (CREATE)

```
CREATE TABLE IF NOT EXISTS Students(ID integer, Name text, Age integer);
```

修改表 (ALTER TABLE)

将Students表改名为Teachers表

```
ALTER TABLE Students RENAME TO Teachers;
```

向Teachers表的结构中，增加一个Sex列

```
ALTER TABLE Teachers ADD COLUMN Sex text;
```

删除表 (DROP TABLE)

```
DROP TABLE Teachers;
```

DML

插入记录 （INSERT INTO）

向Teachers表中插入一条记录

```
INSERT INTO Teachers(id,name,age) values(0,'Dorayo',29);
```

`注意：字符串要加单引号`

删除记录 (DELETE FROME)

从Teachers表中删除指定条件的记录

```
DELETE FROM Teachers WHERE name = 'Dorayo';
```

--> WHERE条件语句的格式

```
关系表达式

WHERE Field = Value
WHERE Field is Value
WHERE Field != Value
WHERE Field is not Value
WHERE Field > Value

逻辑表达式

WHERE Field1 = Value1 and Field2 = Value2
WHERE Field1 = Value1 or Field2 = Value2

注意：以上所有，如果Value是字符串，需要加单引号
```

修改记录 (UPDATE)

更改Teachers表中的指定记录的指定字段

```
UPDATE Teachers SET Age=45,name='Doraemon' WHERE ID=0;
```

DQL

查询记录 (SELECT)

查询Teachers表中指定字段、指定条件的记录

```
SELECT * FROM Teachers;
SELECT Name,Age FROM Teachers;
SELECT Name,Age FROM Teachers WHERE ID != 0;
```

计算记录的数量 (COUNT())

查询Teachers表中，年龄>27的记录的数量

```
SELECT count(Name) FROM Teachers WHERE Age > 27;
SELECT count(*) FROM Teachers WHERE Age > 27;
```

排序 (ORDER BY)

- 查询出来的记录可以用 ORDER BY 来根据字段进行排序

```
SELECT * FROM Teachers ORDER BY Age
```

- 默认是按照升序（由小到大）排序，也可以改为降序排序

```
SELECT * FROM Teachers ORDER BY Age ASC  // 升序

SELECT * FROM Teachers ORDER BY Age DESC  // 降序
```

- 可以在上一个字段的基础上，叠加字段进行排序

```
先按照年龄的升序排序，年龄相同再按照名字的降序排序
SELECT * FROM Teachers ORDER BY Age, Name DESC;
```

限制查询记录数 （LIMIT）

从第3条记录开始，查询显示5条记录

```
SELECT * FROM Teachers LIMIT 3, 5;
```

约束 （Constraints）

什么是约束？ 比如学校教师的教师表，部分字段可能会有如下约束：

```
年龄 - 大于20岁。如果你想录入一个小于20岁的教师，系统会报错
国籍 - 默认中国。所谓默认，就是如果你不填写，系统自动填上默认值
姓名 - 不能为空。如果不填，会报错
员工号 - 唯一。如果重复，会报错
```

> 上面提到的大于、默认、不能为空、唯一等，就是数据的约束条件。在用 CREATE TABLE 创建表的时候，就应该将每个字段列的约束条件事先说明（如果有的话）， 以后再往表里输入数据的时候，系统会自动为我们检查是否满足约束条件，如果不满足，系统会报错。

SQLite 常用约束如下：

- NOT NULL - 非空
- UNIQUE - 唯一
- PRIMARY KEY - 主键
- FOREIGN KEY - 外键
- CHECK - 条件检查
- DEFAULT - 默认

主键 （PRIMARY KEY）

- 数据表中每一条记录都有一个主键，这就像我们每个人的身份证号码、员工号、银行帐号；反过来也可以说，每一个主键对应着一条数据记录。 所以，主键必须是唯一的
- 一般情况下主键同时也是一个索引，所以通过主键查找记录速度比较快

- 在关系型数据库中，一个表的主键可以作为另外一个表的外键， 这样，这两个表之间就通过这个键建立了关系
- 主键一般是整数或者字符串，只要保证唯一就行。 在 SQLite 中，主键如果是整数类型，该列的值可以自动增长

```
sqlite> CREATE TABLE Teachers(ID integer PRIMARY KEY, Name text);
sqlite> .tables
Teachers
sqlite> INSERT INTO Teachers(Name) VALUES('张三');
sqlite> INSERT INTO Teachers(Name) VALUES('李四');
sqlite> INSERT INTO Teachers(Name) VALUES('王五');
sqlite> SELECT * FROM Teachers;
ID          Name
----------  ----------
1           张三
2           李四
3           王五
sqlite> INSERT INTO Teachers(ID,Name) VALUES(2, '田七');
Error: UNIQUE constraint failed: Teachers.ID
```

默认值 （DEFAULT）

有些字段，在每条记录中，值都一样，只在个别情况下才需要更改，可以给这些字段设置默认值

```
sqlite> CREATE TABLE Teachers(ID integer PRIMARY KEY, Name text, Country text DEFAULT '中国');
sqlite> INSERT INTO Teachers(Name) VALUES('张三');
sqlite> INSERT INTO Teachers(Name) VALUES('李四');
sqlite> INSERT INTO Teachers(Name,Country) VALUES('田七','高粱地');
sqlite> SELECT * FROM Teachers;
ID          Name        Country
----------  ----------  ----------
1           张三        中国
2           李四        中国
3           田七        高粱地
sqlite>
```

非空 (NOT NULL)

有些字段，可能一时不知道填什么值，并且也没设置默认值。当添加数据时，这些字段空置不填，系统会认为是NULL

但有另外一些字段，必须被天上值，如果不填，就会出错。这样的字段被称为非空(NOT NULL)字段，需要在定义表的时候事先声明

```
sqlite> DROP TABLE Teachers;
sqlite>
sqlite> CREATE TABLE Teachers(ID integer PRIMARY KEY, Name text, Age integer NOT NULL);
sqlite> INSERT INTO Teachers(Name,Age) VALUES('张三', 23);
sqlite> INSERT INTO Teachers(Name,Age) VALUES('李四', 34);
sqlite> INSERT INTO Teachers(Name) VALUES('王五');
Error: NOT NULL constraint failed: Teachers.Age
sqlite> select * from Teachers;
ID          Name        Age
----------  ----------  ----------
1           张三        23
2           李四        34
sqlite>
```

唯一值 (UNIQUE)

除了主键，可能也有写字段的值不能重复，就需要用到UNIQUE约束了

```
sqlite> DROP TABLE Teachers;
sqlite> .tables
sqlite> CREATE TABLE Teachers(ID integer PRIMARY KEY, Name text UNIQUE);
sqlite> INSERT INTO Teachers(Name) VALUES('Alice');
sqlite> INSERT INTO Teachers(Name) VALUES('Bob');
sqlite> INSERT INTO Teachers(Name) VALUES('Dorayo');
sqlite> INSERT INTO Teachers(Name) VALUES('Bob');
Error: UNIQUE constraint failed: Teachers.Name
sqlite>
```

条件检查 (CHECK)

某些值必须符合一定的条件才允许存入，这是就需要用到这个 CHECK 约束

```
sqlite> CREATE TABLE Teachers(ID integer PRIMARY KEY, Age integer CHECK(Age>22));
sqlite> insert into Teachers(Age) values(38);
sqlite> insert into Teachers(Age) values(49);
sqlite> insert into Teachers(Age) values(20);
Error: CHECK constraint failed: Teachers
sqlite>
```

外键 (FOREIGN KEY)

现在，我们的数据库中已经有 Teachers 表了，假如我们再建立一个 Students 表， 要求 Students 表中的每一个学生都对应一个 Teachers 表中的教师。

很简单，只需要在 Students 表中建立一个 TeacherID 字段，保存对应教师的 ID， 这样，学生和教师之间就建立了关系。

问题是：我们有可能给学生存入一个不在的 Teachers 表中的 TeacherID 值， 而且发现不了这个错误。

这种情况下，可以把 Students 表中 TeacherID 字段声明为一个外键， 让它的值对应到 Teachers 表的 ID 字段上。

这样，一旦在 Students 表中存入一个不存在的教师 Id ，系统就会报错。

```
sqlite> .tables
Teachers
sqlite> create table Students(ID integer PRIMARY KEY, TeacherID integer, FOREIGN KEY(TeacherID) REFERENCES Teachers
(ID));
sqlite> select * from teachers;
ID          Age
----------  ----------
1           38
2           49
sqlite> insert into students(TeacherID) values(1);
sqlite> insert into students(TeacherID) values(2);
sqlite> insert into students(TeacherID) values(3);
sqlite> select * from students;
ID          TeacherID
----------  ----------
1           1
2           2
3           3
sqlite>
```

实际测试中，发现尽管设置了外键约束，依然能够添加不在Teachers表中的ID，比如3。原因是因为外键开关默认是关闭的，打开开关即可使能外键约束。

```
sqlite> PRAGMA foreign_keys = ON;
sqlite> select * from students;
ID          TeacherID
----------  ----------
1           1
2           2
3           3
sqlite> insert into students(TeacherID) values(5);
Error: FOREIGN KEY constraint failed
sqlite>
```

注意：默认情况下，sqlite3动态库中外键功能是关闭的，如果要使用该动态库，需要重新编译该库，并打开响应编译宏开关。详见SQLite Foreign Key Support