一、查看版本信息：

#sqlite3 -version

二、sqlite3常用命令

1、当前目录下建立或打开test.db数据库文件，并进入sqlite命令终端，以sqlite>前缀标识：

2、输出帮助信息：

sqlite>.help

3、查看数据库文件信息命令(注意命令前带字符'.')：

sqlite>.database

4、退出sqlite终端命令：

sqlite>.quit

或

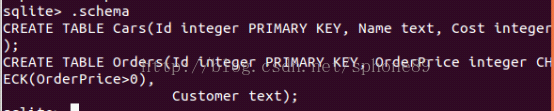
sqlite>.exit

列出当前显示格式的配置：  
sqlite>.show

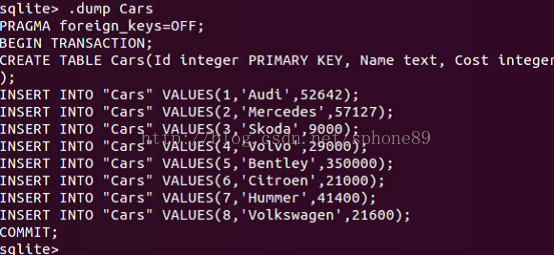
6、显示数据库结构：.schema

显示表的结构：.schema    表名

其实就是一些 SQL 语句，他们描述了数据库的结构，如图



7、导出某个表的数据： .dump    表名



8、设置导出目标：

.output     文件名

或者

.output   stdout

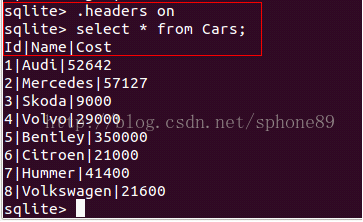
先运行 .output cars.sql ，然后再运行 .dump 命令试试看？如果要回复成导出到终端（标准输出），则运行 .output stdout

10、设置分隔符：.separator    分隔符

我们可以首先运行 SELECT \* FROM Cars； ，可以看到默认的分隔符是 |

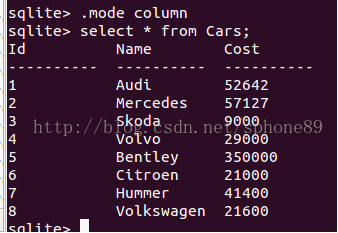
运行.separator : 以后，再 SELECT \* FROM Cars；，可以看到分隔符已经变成 : 了

11、显示标题栏：.headers   on



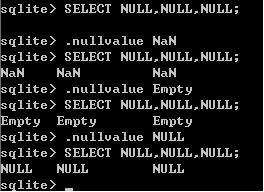
12、设置显示模式：.mode    模式

有好几种显示模式，默认的是 list 显示模式，一般我们使用 column 显示模式，还有其他几种显示模式可以 .help 看 mode 相关内容。看看下面的图，和上面是不是显示的不一样了？



13、设置 NULL 值显示成什么样子： .nullvalue     你想要的NULL值格式

默认情况下NULL值什么也不显示，你可以设置成你自己想要的样子



14、配置文件 .sqliterc

如果我们每次进入命令行都要重新设置显示格式，很麻烦，其实 .show 命令列出的所有设置项都可以保存到一个 .sqliterc 文件中，这样每次进入命令行就自动设置好了。.sqlterc 文件在 Linux 下保存在用户的 Home 目录下，在 Windows 下可以保存到任何目录下，但是需要设置环境变量让数据库引擎能找到它，感兴趣的可以看看帮助。

三、数据库和表的相关命令

1、创建一个新的数据库：sqlite3     文件名

创建一个 test.db 数据库文件，打开控制台窗口，命令如下：

sqlite>sqlite3 test.db

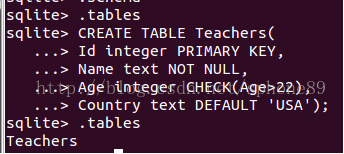
2、打开一个已经存在的数据库：sqlite3 已经存在的文件名

创建一个新数据库和打开一个已经存在的数据库命令是一模一样的，如果文件在当前目录下不存在，则新建；如果存在，则打开。

3、建立数据表

create table table\_name(field1 type1, field2 type1, ...);

table\_name是要创建数据表名称，fieldx是数据表内字段名称，typex则是字段类型。  
如：该语句创建一个记录学生信息的数据表。

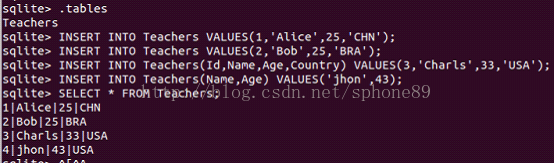


sql的指令格式：所有sql指令都是以分号(;)结尾，两个减号(--)则表示注释。

4、添加数据记录

insert into table\_name(列field1, field2, ...) values(值val1, val2, ...);

valx为需要存入字段的值。  
例，往老师信息表添加数据：

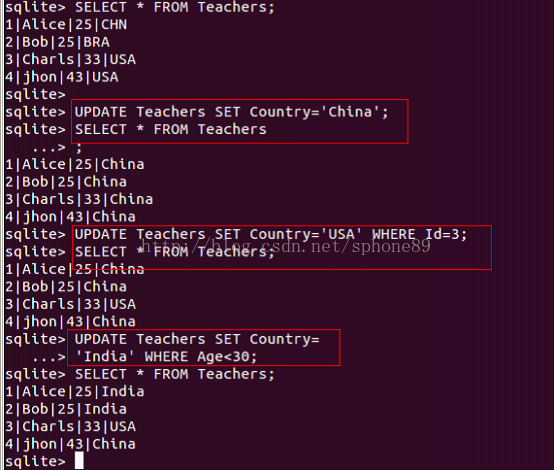


很简单，创建了一个 Teachers 表并向其中添加了四条数据，设定了一些约束，其中有自动增加的主键、默认值等等。

5、修改数据

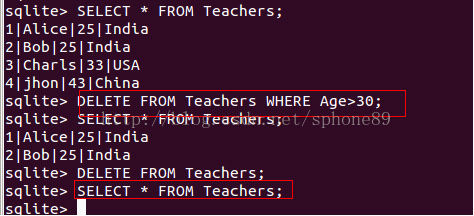
UPDATE 表 SET 列 = '新值' 【WHERE 条件语句】

UPDATE 语句用来更新表中的某个列，如果不设定条件，则所有记录的这一列都被更新； 如果设定了条件，则符合条件的记录的这一列被更新， WHERE 子句被用来设定条件，如下例：



6、删除数据 DELETE FROM 表 【WHERE 条件语句】

如果设定 WHERE 条件子句，则删除符合条件的数据记录；如果没有设定条件语句，则删除所有记录



7、导入数据：.read     数据文件

打开记事本，并将下列 SQL 语句复制到记事本中，保存为 test.sql ，在命令行环境中输入

.read   test.sql

即将所有的数据导入到 test.db 数据库中。

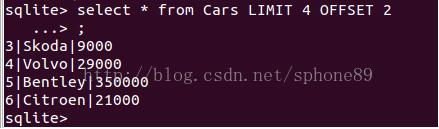
BEGIN TRANSACTION;  
CREATE TABLE Cars(Id integer PRIMARY KEY, Name text, Cost integer);  
INSERT INTO Cars VALUES(1,'Audi',52642);  
INSERT INTO Cars VALUES(2,'Mercedes',57127);  
INSERT INTO Cars VALUES(3,'Skoda',9000);  
INSERT INTO Cars VALUES(4,'Volvo',29000);  
INSERT INTO Cars VALUES(5,'Bentley',350000);  
INSERT INTO Cars VALUES(6,'Citroen',21000);  
INSERT INTO Cars VALUES(7,'Hummer',41400);  
INSERT INTO Cars VALUES(8,'Volkswagen',21600);  
COMMIT;  
  
BEGIN TRANSACTION;  
CREATE TABLE Orders(Id integer PRIMARY KEY, OrderPrice integer CHECK(OrderPrice>0),   
                    Customer text);  
INSERT INTO Orders(OrderPrice, Customer) VALUES(1200, "Williamson");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(200, "Robertson");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(40, "Robertson");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(1640, "Smith");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(100, "Robertson");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(50, "Williamson");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(150, "Smith");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(250, "Smith");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(840, "Brown");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(440, "Black");  
INSERT INTO Orders(OrderPrice, Customer) VALUES(20, "Brown");  
COMMIT;

8、查询数据记录

a查询输出所有数据记录  
select \* from table\_name;

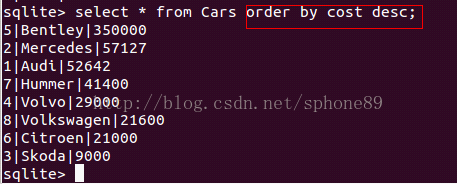
b限制输出数据记录数量

若数据库中的数据太多，全部返回可不行，可以限制返回的数量，还可以设定返回的起始位置  
select \* from table\_name limit val;



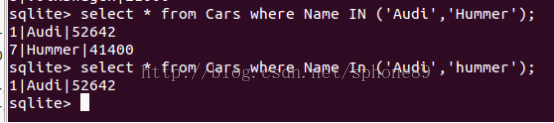
c升序输出数据记录  
select \* from table\_name order by field asc;

d降序输出数据记录  
select \* from table\_name order by field desc;

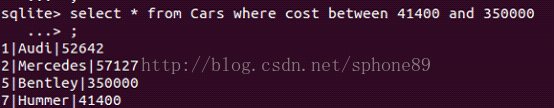


e条件查询  
select \* from table\_name where expression;

IN （集合）  
select \* from table\_name where field in ('val1', 'val2', 'val3');



BETWEEN 值1 AND 值2  
select \* from table\_name where field between val1 and val2;



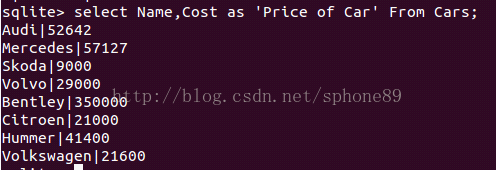
f、查询记录数目

select count (\*) from table\_name;

g、区分列数据  
select distinct field from table\_name;  
有一些字段的值可能会重复出现，distinct去掉重复项，将列中各字段值单个列出。

h、别名 SELECT 列 AS 别名，列 AS 别名 FROM

可以给返回数据集中的某些列起一个比较直观的名字，比如把 Cost 改为"Price Of Car"



I、条件查询 SELECT 列 FROM 表 【WHERE 条件语句】

一般的条件语句都是大于、小于、等于之类的，这里有几个特别的条件语句

LIKE

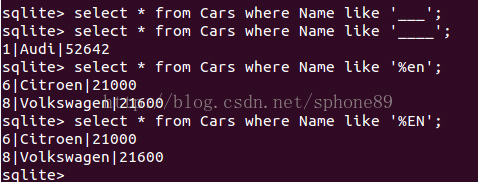
------------------------------

LIKE 用通配符匹配字符串

下划线 \_ 匹配一个字符串

百分号 % 匹配多个字符串

LIKE 匹配字符串时不区分大小写

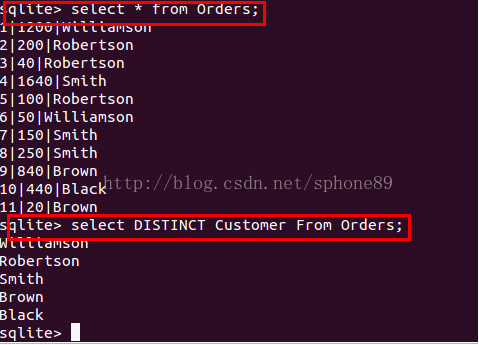


GLOB

J 区分 DISTINCT 列

有一些字段的值可能会出现重复，比如订单表中，一个客户可能会有好几份订单，因此客户的名字会重复出现。

到底有哪些客户下了订单呢？下面的语句将客户名字区分出来。

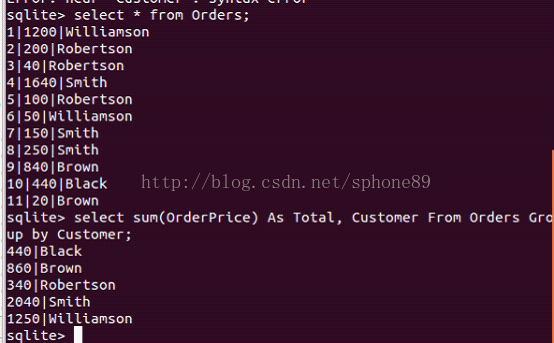


k分组 GROUP BY 列

分组和前面的区分有一点类似。区分仅仅是为了去掉重复项，而分组是为了对各类不同项进行统计计算。

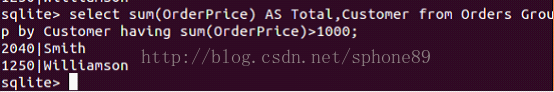
比如上面的例子，我们区分出 5 个客户，这 5 个客户一共下了 11 个订单，说明很多客户都下了不止一个订单。

下面的语句统计每个客户在订单上总共花费了多少钱。



这里 Sum 是 SQLite 内置的统计函数，在这个例子中用来求每个顾客的订单价格的和。

统计结果也可以设定返回条件，但是不能用 WHERE 子句，而是用 HAVING 子句，如下例，返回订单总额大于 1000 的顾客。



9、建立索引  
当说数据表存在大量记录，索引有助于加快查找数据表速度。  
create index index\_name on table\_name(field);  
例，针对学生表stu\_no字段，建立一个索引：  
create index student\_index on student\_table(stu\_no);  
建立完成后，sqlite3在对该字段查询时，会自动使用该索引。  
   
10、删除数据表或索引  
drop table table\_name;  
drop index index\_name;

四、sqlite3存储数据的类型  
NULL：标识一个NULL值  
INTERGER：整数类型  
REAL：浮点数  
TEXT：字符串  
BLOB：二进制数

五、sqlite3存储数据的约束条件  
Sqlite常用约束条件如下：  
PRIMARY KEY - 主键：  
1）主键的值必须唯一，用于标识每一条记录，如学生的学号  
2）主键同时也是一个索引，通过主键查找记录速度较快  
3）主键如果是整数类型，该列的值可以自动增长  
NOT NULL - 非空：  
约束列记录不能为空，否则报错  
UNIQUE - 唯一：  
除主键外，约束其他列的数据的值唯一  
CHECK - 条件检查：  
约束该列的值必须符合条件才可存入  
DEFAULT - 默认值：  
列数据中的值基本都是一样的，这样的字段列可设为默认值

CREATE TABLE Orders(Id integer PRIMARY KEY, OrderPrice integer CHECK(OrderPrice>**0**),   
                    Customer text);

CREATE TABLE Friends(Id integer PRIMARY KEY, Name text UNIQUE NOT NULL,   
                     Sex text CHECK(Sex IN ('M', 'F')));

CREATE TABLE IF NOT EXISTS Reservations(Id integer PRIMARY KEY,   
                                        CustomerId integer, Day text);  
INSERT INTO Reservations(CustomerId, Day) VALUES(**1**, '2009-22-11');

CREATE TABLE Books(Id integer PRIMARY KEY, Title text, Author text,   
                   Isbn text default 'not available');

**怎么在已经创建的表中插入一列**

ALTER TABLE table-name

ADD COLUMN column-name column-type