

RSS 订阅

1/13

sqlite3常用命令&语法	(69052)
java中的final关键字所起	(11063)
websocket	(8180)
socket.io.emit 方法	(7547)
excel中计算两个日期之间的天数	(6005)
android:layout_width="n"	(4541)
用iframe实现不刷新整个页面	(4484)
android中的闹钟管理器AlarmManager	(2740)
struts2数据校验	(2590)
struts2类型转换器	(2192)

评论排行	
java中的final关键字所起	(3)
sqlite3常用命令&语法	(3)
websocket	(2)
perl的helloworld	(0)
java中assert基本使用	(0)
用iframe实现不刷新整个页面	(0)
android中的闹钟管理器AlarmManager	(0)
MYSQL常用语法	(0)
django实现登录和注册之django.contrib.auth	(0)
rokon中的碰撞检测	(0)

推荐文章	
* 致JavaScript也将征服的物联网世界	
* 从苏宁电器到卡巴斯基：难忘的三年硕士时光	
* 作为一名基层管理者如何利用情商管理自己和团队（一）	
* Android CircleImageView圆形ImageView	
* 高质量代码的命名法则	

最新评论	
java中的final关键字所起的作用	梓明: 类似就是最终的变量或者属性 不能够被修改
java中的final关键字所起的作用	no_sunday: 有帮助，谢谢博主
sqlite3常用命令&语法	leidengyan: 很棒
java中的final关键字所起的作用	请叫我石臻臻臻臻: 不错 但是总结的不全
sqlite3常用命令&语法	yaoelvon: 学习中，文章显凌乱，没有明显的条理。希望博主整理下
websocket	hao362980633: 请问我怎么打开连接的时候判断是否建立这个连接。client建立连接的时候可以传一些参数吗
websocket	kaikai5566: 学习了
sqlite3常用命令&语法	tangxingyou: 路过，学习

.schema databaseobjectname 查看创建该数据库对象时的SQL的命令：如果没有这个数据库对象就不显示内容，不会有错误提示

.read FILENAME 执行指定文件中的SQL语句
.headers on/off 显示表头 默认off

.mode list|column|insert|line|tabs|tcl|csv 改变输出格式，具体如下

```
sqlite> .mode list
sqlite> select * from emp;
7369|SMITH|CLERK|7902|17-12-1980|800||20
7499|ALLEN|SALESMAN|7698|20-02-1981|1600|300|30
如果字段值为NULL 默认不显示 也就是显示空字符串
```

```
sqlite> .mode column
sqlite> select * from emp;
7369      SMITH      CLERK      7902      17-12-1980  800              20
7499      ALLEN      SALESMAN   7698      20-02-1981  1600      300      30
7521      WARD      SALESMAN   7698      22-02-1981  1250      500      30
```

```
sqlite> .mode insert
sqlite> select * from dept;
INSERT INTO table VALUES(10,'ACCOUNTING','NEW YORK');
INSERT INTO table VALUES(20,'RESEARCH','DALLAS');
INSERT INTO table VALUES(30,'SALES','CHICAGO');
INSERT INTO table VALUES(40,'OPERATIONS','BOSTON');
```

```
sqlite> .mode line
sqlite> select * from dept;
DEPTNO = 10
DNAME = ACCOUNTING
LOC = NEW YORK
```

```
DEPTNO = 20
DNAME = RESEARCH
LOC = DALLAS
```

```
DEPTNO = 30
DNAME = SALES
LOC = CHICAGO
```

```
DEPTNO = 40
DNAME = OPERATIONS
LOC = BOSTON
```

快速回复

返回顶部



```
sqlite> .mode tabs
sqlite> select * from dept;
10 ACCOUNTING  NEW YORK
20 RESEARCH   DALLAS
30 SALES      CHICAGO
40 OPERATIONS BOSTON

sqlite> .mode tcl
sqlite> select * from dept;
"10" "ACCOUNTING""NEW YORK"
"20" "RESEARCH""DALLAS"
"30" "SALES" "CHICAGO"
"40" "OPERATIONS""BOSTON"
```

```
sqlite> .mode csv
sqlite> select * from dept;
10,ACCOUNTING,"NEW YORK"
20,RESEARCH,DALLAS
30,SALES,CHICAGO
40,OPERATIONS,BOSTON
```

```
.separator "X" 更改分界符号为X
sqlite> .separator '*'
sqlite> select * from dept;
10**ACCOUNTING**NEW YORK"
20**RESEARCH**DALLAS
30**SALES**CHICAGO
40**OPERATIONS**BOSTON
```

- .dump ?TABLE? 生成形成数据库表的SQL脚本
- .dump 生成整个数据库的脚本在终端显示
- .output stdout 将输出打印到屏幕 默认
- .output filename 将输出打印到文件（.dump .output 结合可将数据库以sql语句的形式导出到文件中）
- .nullvalue STRING 查询时用指定的串代替输出的NULL串 默认为.nullvalue "

字段类型：

- 数据库中存储的每个值都有一个类型,都属于下面所列类型中的一种,(被数据库引擎所控制)
- NULL: 这个值为空值
- INTEGER: 值被标识为整数,依据值的大小可以依次被存储为1,2,3,4,5,6,7,8个字节
- REAL: 所有值都是浮动的数值,被存储为8字节的IEEE浮点标记序号.
- TEXT: 文本. 值为文本字符串,使用数据库编码存储(TUTF-8, UTF-16BE or UTF-16LE)
- BLOB: 值是BLOB数据,如何输入就如何存储,不改变格式.

值被定义为什么类型只和值自身有关,和列没有关系,和变量也没有关系
数据库引擎将在执行时检查、解析类型，并进行数字存储类型(整数

led灯价格

快速回复

返回顶部

关闭

5W
7W
9W
12W
15W

SQL语句中部分的带双引号或单引号的文字被定义为文本，
如果文字没带引号并没有小数点或指数则被定义为整数，
如果文字没带引号但有小数点或指数则被定义为实数，
如果值是空则被定义为空值。
BLOB数据使用符号X'ABCD'来标识。



但实际上，sqlite3也接受如下的数据类型：

smallint 16位的整数。

integer 32位的整数。

decimal(p,s) 精确值p是指全部有几个十进制数,s是指小数点后可以有几位小数。如果没有特别指定，则系统会默认为p=5 s=0。

float 32位元的实数。

double 64位元的实数。

char(n) n 长度的字串，n不能超过 254。

varchar(n) 长度不固定且其最大长度为 n 的字串，n不能超过 4000。

graphic(n) 和 **char(n)** 一样，不过其单位是两个字节，n不能超过127。这个形态是为了支持两个字节长度的字体，如中文字。

vargraphic(n) 可变长度且其最大长度为n的双字节元字串，n不能超过2000

date 包含了 年份、月份、日期。

time 包含了 小时、分钟、秒。

timestamp 包含了 年、月、日、时、分、秒、千分之一秒。

快速回复

返回顶部

SQLite包含了如下时间/日期函数：

datetime() 产生日期和时间 无参数表示获得当前时间和日期

sqlite> select datetime();

2012-01-07 12:01:32

有字符串参数则把字符串转换成日期

sqlite> select datetime('2012-01-07 12:01:30');

2012-01-07 12:01:30

select date('2012-01-08','+1 day','+1 year');

2013-01-09

select datetime('2012-01-08 00:20:00','+1 hour','-12 minute');

2012-01-08 01:08:00

select datetime('now','start of year');

2012-01-01 00:00:00

select datetime('now','start of month');

2012-01-01 00:00:00

select datetime('now','start of day');

2012-01-08 00:00:00

select datetime('now','start of week');错误

关闭



5W
7W
9W
12W
15W

```
select datetime('now','localtime');
结果: 2006-10-17 21:21:47
```

```
date()产生日期
sqlite> select date('2012-01-07 12:01:30');
2012-01-07
同理 有参和无参
select date('now','start of year');
2012-01-01
```

```
select date('2012-01-08','+1 month');
2012-02-08
```

```
time() 产生时间
select time();
03:14:30
```

```
select time('23:18:59');
23:18:59
```

```
select time('23:18:59','start of day');
00:00:00
```

```
select time('23:18:59','end of day');错误
```

在时间/日期函数里可以使用如下格式的字符串作为参数:

- YYYY-MM-DD
- YYYY-MM-DD HH:MM
- YYYY-MM-DD HH:MM:SS
- YYYY-MM-DD HH:MM:SS.SSS
- HH:MM
- HH:MM:SS
- HH:MM:SS.SSS
- now








其中now是产生现在的时间。

日期不能正确比较大小,会按字符串比较, 日期默认格式 dd-mm-yyyy

```
select hiredate from emp order by hiredate;
```


```
17-11-1981
17-12-1980
19-04-1987
20-02-1981
```

led灯价格



快速回复

返回顶部



5W
7W
9W
12W
15W

22-02-1981



`strftime()` 对以上三个函数产生的日期和时间进行格式化
`strftime()`函数可以把YYYY-MM-DD HH:MM:SS格式的日期字符串转换成其它形式的字符串。 `strftime(格式, 日期/时间, 修正符, 修正符, ...)` `select strftime('%d',datetime());`
它可以用以下的符号对日期和时间进行格式化:
`%d` 在该月中的第几天, 01-31
`%f` 小数形式的秒, SS.SSS
`%H` 小时, 00-23
`%j` 算出某一天是该年的第几天, 001-366
`%m` 月份, 00-12
`%M` 分钟, 00-59
`%s` 从1970年1月1日到现在的秒数
`%S` 秒, 00-59
`%w` 星期, 0-6 (0是星期天)
`%W` 算出某一天属于该年的第几周, 01-53
`%Y` 年, YYYY
`%%` 百分号

快速回复

返回顶部

```
select strftime('%Y.%m.%d %H:%M:%S','now');
select strftime('%Y.%m.%d %H:%M:%S','now','localtime');
```

结果: 2006.10.17 21:41:09

```
select hiredate from emp
order by strftime('%Y.%m.%d %H:%M:%S',hiredate); 正确
```

```
select strftime('%Y.%m.%d %H:%M:%S',hiredate) from emp
order by strftime('%Y.%m.%d %H:%M:%S',hiredate); 错误
```

算术函数

`abs(X)` 返回给定数字表达式的绝对值。
`max(X,Y[,...])` 返回表达式的最大值。 组函数 `max(列名)`
`sqlite> select max(2,3,4,5,6,7,12);`
12

`min(X,Y[,...])` 返回表达式的最小值。
`random()` 返回随机数。
`sqlite> select random();`
3224224213599993831

`round(X[,Y])` 返回数字表达式并四舍五入为指定的长度或精度。

字符处理函数

`length(X)` 返回给定字符串表达式的字符个数。
`lower(X)` 将大写字符数据转换为小写字符数据后返回字符表达式。



upper(X) 返回将小写字符数据转换为大写的字符表达式。
substr(X,Y,Z) 返回表达式的一部分。 从Y开始读Z个字符 Y最小值:
sqlite> select substr('abcdef',3,3);
cde

quote(A) 给字符串加引号
sqlite> select quote('aaa');
'aaa'

条件判断函数
ifnull(X,Y) 如果X为null 返回Y
select ifnull(comm,0) from emp;
0
300
500
0
1400


集合函数
avg(X) 返回组中值的平均值。
count(X) 返回组中项目的数量。
max(X) 返回组中值的最大值。
min(X) 返回组中值的最小值。
sum(X) 返回表达式中所有值的和。

其他函数
typeof(X) 返回数据的类型。
sqlite> select typeof(111);
integer
sqlite> select typeof('233');
text
sqlite> select typeof('2012-12-12');
text
sqlite> select typeof('223.44');
text
sqlite> select typeof(223.44);
real

last_insert_rowid() 返回最后插入的数据的ID。
sqlite_version() 返回SQLite的版本。
sqlite> select sqlite_version();
3.7.9

change_count() 返回受上一语句影响的行数。
last_statement_change_count()

led灯价格




快速回复

返回顶部

关闭

✕



5W
7W
9W
12W
15W

create table emp_bak select * from EMP;不能在sqlite中使用

插入记录

insert into table_name values (field1, field2, field3...);

查询

select * from table_name;查看table_name表中所有记录;

select * from table_name where field1='xxxxx'; 查询符合指定条件的记录;

select

from table_name[,table_name2,...]

where

group by....

having

order by ...

select

from table_name inner join | left outer join | right outer join table_name2

on ...

where

group by....

having

order by ...

子查询:

select *

from EMP m

where SAL>

(select avg(SAL) from EMP where DEPTNO=m.DEPTNO);

支持case when then 语法

update EMP

set SAL=

(

case

when DEPTNO=10 and JOB='MANAGER' then SAL*1.1

when DEPTNO=20 and JOB='CLERK' then SAL*1.2

when DEPTNO=30 then SAL*1.1

when DEPTNO=40 then SAL*1.2

else SAL

END

);

select ENAME,

case DEPTNO

when 10 then '后勤部'

when 20 then '财务部'

when 30 then '内务部门'



快速回复

返回顶部

关闭




```
else '其他部门'  
end as dept  
from EMP;
```



支持关联子查询 in后面的语法中可以有limit (mysql不可以)

```
select *  
from emp e  
where e.EMPNO in  
(  
select empno  
from EMP  
where deptno=e.DEPTNO  
order by SAL desc  
limit 0,2  
);
```

快速回复

返回顶部

支持表和表之间的数据合并等操作

union 去重复 union all 不去掉重复

```
select deptno from emp  
union  
select deptno from dept;
```

```
select deptno from emp  
union all  
select deptno from dept;
```

在列名前加distinct也是去重复

```
sqlite> select distinct deptno from emp;
```

删除

```
delete from table_name where ...
```

删除表

```
drop table_name; 删除表;  
drop index_name; 删除索引;
```

修改

```
update table_name  
set xxx=value[, xxx=value,...]  
where ...
```

建立索引



如果资料表有相当多的资料，我们便会建立索引来加快速度。好比说：



```
create index film_title_index on film(title);
```

意思是针对film资料表的name字段，建立一个名叫film_name_index的索引。这个指令的语法为

```
CREATE [ UNIQUE ] NONCLUSTERED INDEX index_name
ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
```

```
create index index_name on table_name(field_to_be_indexed);
```

一旦建立了索引，sqlite3会在针对该字段作查询时，自动使用该索引。这一切的操作都是在幕后自动发生的，无须使用者特别指令。

快速回复

返回顶部

其他sqlite的特别用法

```
sqlite可以在shell底下直接执行命令：
sqlite3 film.db "select * from emp;"
```

```
输出 HTML 表格：
sqlite3 -html film.db "select * from film;"
将数据库「倒出来」：
```

```
sqlite3 film.db ".dump" > output.sql
利用输出的资料，建立一个一模一样的数据库（加上以上指令，就是标准的SQL数据库备份了）：
```

```
sqlite3 film.db < output.sql
在大量插入资料时，你可能会需要先打这个指令：
```

```
begin;
插入完资料后要记得打这个指令，资料才会写进数据库中：
commit;
```

```
sqlite> begin;
sqlite> insert into aaaa values('aaa','333');
sqlite> select * from aaaa;
2|sdfds
sdfsd|9
2012-12-12|13:13:13
aaa|333
sqlite> rollback;
sqlite> select * from aaaa;
2|sdfds
sdfsd|9
2012-12-12|13:13:13
```

创建和删除视图



```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
DROP VIEW view_name
```

```
create view e as
select avg(SAL) avgsal,DEPTNO
from EMP
group by DEPTNO;
```

```
select ENAME,EMP.DEPTNO,SAL,avgsal
from EMP inner join e
on EMP.DEPTNO=e.DEPTNO
where SAL>avgsal;
```

练习员工表:

```
PRAGMA foreign_keys=OFF;
```

```
BEGIN TRANSACTION;
```

```
CREATE TABLE DEPT
```

```
(
DEPTNO int(2) not null,
DNAME varchar(14),
LOC varchar(13)
);
```

```
INSERT INTO "DEPT" VALUES(10,'ACCOUNTING','NEW YORK');
```

```
INSERT INTO "DEPT" VALUES(20,'RESEARCH','DALLAS');
```

```
INSERT INTO "DEPT" VALUES(30,'SALES','CHICAGO');
```

```
INSERT INTO "DEPT" VALUES(40,'OPERATIONS','BOSTON');
```

```
CREATE TABLE EMP
```

```
(
EMPNO int(4) not null,
ENAME varchar(10),
JOB varchar(9),
MGR int(4),
HIREDATE date,
SAL int(7),
COMM int(7),
DEPTNO int(2)
);
```

```
INSERT INTO "EMP" VALUES(7369,'SMITH','CLERK',7902,'17-12-1980',800,NULL,20);
```

```
INSERT INTO "EMP" VALUES(7499,'ALLEN','SALESMAN',7698,'20-02-1981',1600,300,30);
```

```
INSERT INTO "EMP" VALUES(7521,'WARD','SALESMAN',7698,'22-02-1981',1250,500,30);
```

```
INSERT INTO "EMP" VALUES(7566,'JONES','MANAGER',7839,'02-04-1981',2975,NULL,20);
```

```
INSERT INTO "EMP" VALUES(7654,'MARTIN','SALESMAN',7698,'28-09-1981',1250,1400,30);
```

```
INSERT INTO "EMP" VALUES(7698,'BLAKE','MANAGER',7839,'01-05-1981',2850,NULL,30);
```

```
INSERT INTO "EMP" VALUES(7782,'CLARK','MANAGER',7839,'09-06-1981',2450,NULL,10);
```

```
INSERT INTO "EMP" VALUES(7788,'SCOTT','ANALYST',7566,'19-04-1987',3000,NULL,20);
```

```
INSERT INTO "EMP" VALUES(7839,'KING','PRESIDENT',NULL,'17-11-1981',5000,0,10);
```

```
INSERT INTO "EMP" VALUES(7844,'TURNER','SALESMAN',7698,'08-09-1981',1500,0,30);
```

```
INSERT INTO "EMP" VALUES(7876,'ADAMS','CLERK',7788,'23-08-1981',1200,0,20);
```

```
INSERT INTO "EMP" VALUES(7900,'JAMES','CLERK',7698,'03-12-1981',950,0,20);
```

```
INSERT INTO "EMP" VALUES(7902,'FORD','ANALYST',7566,'03-12-1981',3000,0,20);
```



快速回复

返回顶部

关闭



```
INSERT INTO "EMP" VALUES(7934,'MILLER','CLERK',7782,'23-09-1980');
CREATE TABLE SALGRADE
(
  GRADE int,
  LOSAL int,
  HISAL int
);
INSERT INTO "SALGRADE" VALUES(1,700,1200);
INSERT INTO "SALGRADE" VALUES(2,1201,1400);
INSERT INTO "SALGRADE" VALUES(3,1401,2000);
INSERT INTO "SALGRADE" VALUES(4,2001,3000);
INSERT INTO "SALGRADE" VALUES(5,3001,9999);
COMMIT;
```



快速回复

返回顶部



顶 1 踩 1

上一篇 android:layout_width="match_parent"
下一篇 MYSQL常用语法

我的同类文章

android (2)	
• android开发转载 2013-03-24 阅读 360	• android中的闹钟管理器Ala... 2012-02-08 阅读 2739

猜你在找

- 《C语言/C++学习指南》数据库篇(MySQL& sqlite)
- sqlite3常用命令&语法
- iOS开发高级专题—数据存储
- sqlite3常用命令&语法
- 360度解析亚马逊AWS数据存储服务
- Linux命令汇总
- SQL Server 2005数据库高级应用
- Linux命令汇总
- Oracle数据库从入门到精通
- Linux SVN安装



查看评论

3楼 leidengyan 2014-09-25 14:27发表



很棒

2楼 yaoelvon 2014-01-14 16:39发表



学习中，文章显凌乱，没有明显的条理。希望博主整理下

1楼 tangxingyou 2012-02-24 17:02发表



路过，学习

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目



全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

Java

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

LBS

Unity

Splashtop

UML

components

Windows Mobile

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap

led灯价格



关闭 X



5W
7W
9W
12W
15W