SQLite是Android自带的轻量级数据库

SQLite 是不区分大小写的，但也有一些命令是大小写敏感的，比如 GLOB 和 glob 在 SQLite 的语句中有不同的含义。

所有的 SQLite 语句可以以任何关键字开始，如 SELECT、INSERT、UPDATE、DELETE、ALTER、DROP 等，所有的语句以分号结束。

SQLite可以使用 C 风格的注释，以 "/\*" 开始，至下一个 "\*/" 字符对或直到输入结束

SQL

用于访问和处理数据库的标准的计算机语言。 Structured Query Language。

RDBMS 指关系型数据库管理系统，全称 Relational Database Management System。

SQL 对大小写不敏感

一些最重要的 SQL 命令

SELECT - 从数据库中提取数据

UPDATE - 更新数据库中的数据

DELETE - 从数据库中删除数据

INSERT INTO - 向数据库中插入新数据

CREATE DATABASE - 创建新数据库

ALTER DATABASE - 修改数据库

CREATE TABLE - 创建新表

ALTER TABLE - 变更（改变）数据库表

DROP TABLE - 删除表

CREATE INDEX - 创建索引（搜索键）

DROP INDEX - 删除索引

SQL SELECT DISTINCT 语句

在表中，一个列可能会包含多个重复值，有时您也许希望仅仅列出不同（distinct）的值。

DISTINCT 关键词用于返回唯一不同的值。

SQL WHERE 子句

WHERE 子句用于提取那些满足指定标准的记录。

SELECT column\_name,column\_name

FROM table\_name

WHERE column\_name operator value;

SQL 使用单引号来环绕文本值（大部分数据库系统也接受双引号）。

如果是数值字段，请不要使用引号，例如SELECT \* FROM Websites WHERE id=1;

SQL的运算符operator



SQL AND & OR 运算符

用于基于一个以上的条件对记录进行过滤。

您也可以把 AND 和 OR 结合起来（使用圆括号来组成复杂的表达式）。

例如：SELECT \* FROM Websites

WHERE alexa > 15

AND (country='CN' OR country='USA');

SQL ORDER BY 关键字

用于对结果集按照一个列或者多个列进行排序。如果要降序排序，可使用 DESC 关键字。

ORDER BY column\_name,column\_name ASC|DESC; （默认是升序）

ORDER BY 多列

ORDER BY country,alexa; （按照country从小到大排，遇到country相等的情况就按alexa排）

SQL INSERT INTO 语句

用于向表中插入新记录。可以有两种编写形式。

第一种形式无需指定要插入数据的列名，只需提供被插入的值即可：

INSERT INTO table\_name

VALUES (value1,value2,value3,...);

第二种形式需要指定列名及被插入的值：

INSERT INTO table\_name (column1,column2,column3,...)

VALUES (value1,value2,value3,...);

SQL UPDATE 语句

用于更新表中的记录。

SQL UPDATE 语法

UPDATE table\_name

SET column1=value1,column2=value2,...

WHERE some\_column=some\_value;

注意：如果您省略了 WHERE 子句，所有的记录都将被更新！执行没有 WHERE 子句的 UPDATE 要慎重，再慎重。 MySQL 中可以通过设置 sql\_safe\_updates 这个自带的参数来解决，当该参数开启的情况下，你必须在update 语句后携带 where 条件，否则就会报错。

set sql\_safe\_updates=1; 表示开启该参数

SQL DELETE 语句

用于删除表中的记录。

SQL DELETE 语法

DELETE FROM table\_name

WHERE some\_column=some\_value;

注意：如果您省略了 WHERE 子句，所有的记录都将被删除！

您可以在不删除表的情况下，删除表中所有的行。这意味着表结构、属性、索引将保持不变：

DELETE FROM table\_name;

或

DELETE \* FROM table\_name;

注释：在删除记录时要格外小心！因为您不能重来！

DROP、TRUNCATE、 DELETE 的区别。

DROP test;（删除表test，并释放空间，将test删除的一干二净）

TRUNCATE test; （删除表test里的内容，并释放空间，保留表的定义，表的结构）

DELETE: 1、删除指定数据 2、删除整个表（仅删除表test内的所有内容，保留表的定义和结构，不释放空间）

delete 命令删除的数据将存储在系统回滚段中，需要的时候，数据可以回滚恢复，而 truncate 命令删除的数据是不可以恢复的。

小心使用 drop 和 truncate, 尤其没有备份的时候。

总结：除非删除表用drop，否则尽量都用delete。

SQL SELECT LIMIT 子句

用于规定要返回的最前面的记录的数目。对于拥有数千条记录的大型表来说是非常有用的。

SELECT column\_name(s)

FROM table\_name

LIMIT number;

SQL LIKE 操作符

LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式。

SQL LIKE 语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name LIKE pattern;

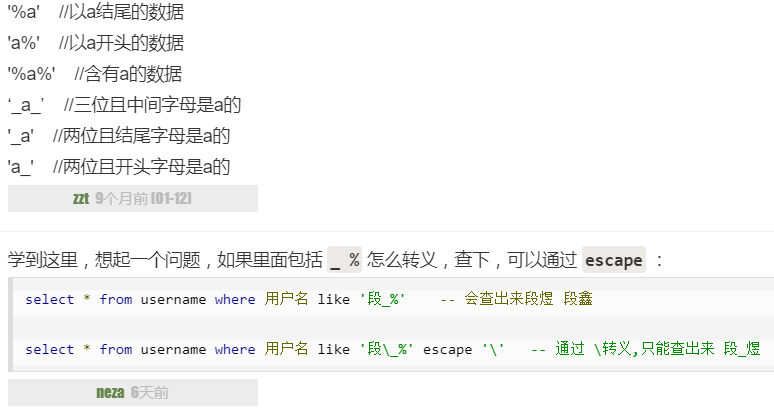
SELECT \* FROM Websites

WHERE name LIKE '%k';

通过使用 NOT 关键字，您可以选取不匹配模式的记录。

例如：SELECT \* FROM Websites

WHERE name NOT LIKE '%oo%';



SQL 通配符

在 SQL 中，通配符与 SQL LIKE 操作符一起使用。通配符可替代字符串中的任何其他字符。



使用 SQL [charlist] 通配符

MySQL中使用 REGEXP 或 NOT REGEXP 运算符 (或 RLIKE和NOT RLIKE) 来操作正则表达式。

SQL IN 操作符

允许您在 WHERE 子句中规定多个值。

SQL IN 语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name IN (value1,value2,...); （可以转换成=加上OR的表达式）

SQL BETWEEN 操作符

用于选取介于两个值之间的数据范围内的值。这些值可以是数值、文本或者日期。

BETWEEN '2016-05-10' AND '2016-05-14';

如需显示不在上面实例范围内的网站，请使用 NOT BETWEEN：

SELECT \* FROM Websites

WHERE name NOT BETWEEN 'A' AND 'H';

注意：在不同的数据库中，BETWEEN的边界是不同的（左右的开或闭）

SQL 别名

用AS关键字可以为表名称或列名称指定别名。

列的 SQL 别名语法

SELECT column\_name AS alias\_name

FROM table\_name;

表的 SQL 别名语法

SELECT column\_name(s)

FROM table\_name AS alias\_name;

提示：如果列名称包含空格，要求使用双引号或方括号。



在下面的情况下，使用别名很有用：

在查询中涉及超过一个表

在查询中使用了函数

列名称很长或者可读性差

需要把两个列或者多个列结合在一起

SQL 连接(JOIN)

用于把来自两个或多个表的行结合起来。

最常见的 JOIN 类型：SQL INNER JOIN（默认的JOIN即为这个）

SELECT column\_name(s)

FROM table1

INNER JOIN table2

ON table1.column\_name=table2.column\_name;

注意：ON和WHERE的区别，on 条件是在生成临时表时使用的条件，它不管 on 中的条件是否为真，都会返回左边表中的记录。where 条件是在临时表生成好后，再对临时表进行过滤的条件。这时已经没有 left join 的含义（必须返回左边表的记录）了，条件不为真的就全部过滤掉。如果是inner join，条件放在on中和where中就没有差别。

参考网址：<http://www.runoob.com/w3cnote/sql-join-the-different-of-on-and-where.html>

其他类型的JOIN：

LEFT JOIN 关键字从左表（Websites）返回所有的行，即使右表（access\_log）中没有匹配。

RIGHT JOIN 关键字从右表（Websites）返回所有的行，即使左表（access\_log）中没有匹配。

~~FULL OUTER JOIN 关键字，等于LEFT JOIN和RIGHT JOIN的合并，MySQL不支持~~

SQL UNION 操作符

用于合并两个或多个 SELECT 语句的结果集。

注意：UNION 内部的每个 SELECT 语句必须拥有相同数量的列。列也必须拥有相似的数据类型。UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。

SQL CREATE DATABASE 语句

用于创建数据库。 CREATE DATABASE dbname;

SQL CREATE TABLE 语句

用于创建数据库中的表。

SQL CREATE TABLE 语法

CREATE TABLE table\_name

(

column\_name1 data\_type(size),

column\_name2 data\_type(size),

column\_name3 data\_type(size),

....

);

column\_name 参数规定表中列的名称。

data\_type 参数规定列的数据类型（例如 varchar、integer、decimal、date 等等）。

size 参数规定表中列的最大长度。

SQL数据类型参考：<http://www.runoob.com/sql/sql-datatypes.html>

在 INSERT 或 UPDATE 查询中，TIMESTAMP 自动把自身设置为当前的日期和时间。

SQL 约束（Constraints）

用于规定表中的数据规则。

如果存在违反约束的数据行为，行为会被约束终止。

约束可以在创建表时规定（通过 CREATE TABLE 语句），或者在表创建之后规定（通过 ALTER TABLE 语句）。

column\_name1 data\_type(size) constraint\_name

在 SQL 中，我们有如下约束：

NOT NULL - 指示某列不能存储 NULL 值。这意味着，如果不向字段添加值，就无法插入新记录或者更新记录。

UNIQUE - 保证某列的每行必须有唯一的值。PRIMARY KEY 约束拥有自动定义的 UNIQUE 约束。每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束。

PRIMARY KEY - NOT NULL和UNIQUE的结合。确保某列（或两个列多个列的结合）有唯一标识，有助于更容易更快速地找到表中的一个特定的记录。每个表都应该有一个主键，并且每个表只能有一个主键。（注意：一个主键可能是由两个或更多列组成的）

FOREIGN KEY - 保证一个表中的数据匹配另一个表中的值的参照完整性。一个表中的 FOREIGN KEY 指向另一个表中的 UNIQUE KEY(唯一约束的键)。FOREIGN KEY 约束用于预防破坏表之间连接的行为。FOREIGN KEY 约束也能防止非法数据插入外键列，因为它必须是它指向的那个表中的值之一。

CHECK - 保证列中的值符合指定的条件。

DEFAULT - 规定没有给列赋值时的默认值。

如需定义多个列的 UNIQUE 约束，可以使用：

CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName) //uc\_PersonID是约束的名字

当表已被创建时，可以这样定义约束：

ALTER TABLE Persons

ADD UNIQUE (P\_Id) 或者 ADD CONSTRAINT uc\_PersonID UNIQUE (P\_Id,LastName)

如需撤销 UNIQUE 约束，可以使用：

ALTER TABLE Persons

DROP INDEX uc\_PersonID

如需撤销 PRIMARY KEY 约束，可以使用：

ALTER TABLE Persons

DROP PRIMARY KEY

表创建时建立外键约束：FOREIGN KEY (P\_Id) REFERENCES Persons(P\_Id)

表已创建时添加外键约束：

ALTER TABLE Orders

ADD FOREIGN KEY (P\_Id)

REFERENCES Persons(P\_Id)

如需撤销 FOREIGN KEY 约束，请使用下面的 SQL：

ALTER TABLE Orders

DROP FOREIGN KEY fk\_PerOrders

SQL CHECK 约束

用于限制列中的值的范围。

CHECK (P\_Id>0) 或 CONSTRAINT chk\_Person CHECK (P\_Id>0 AND City='Sandnes')

表已创建：ALTER TABLE Persons

ADD CONSTRAINT chk\_Person CHECK (P\_Id>0 AND City='Sandnes')

撤销：ALTER TABLE Persons

DROP CHECK chk\_Person

SQL DEFAULT 约束

用于向列中插入默认值。

表已创建：ALTER TABLE Persons

ALTER City SET DEFAULT 'SANDNES'

撤销：ALTER TABLE Persons

ALTER City DROP DEFAULT

SQL CREATE INDEX 语句

用于在表中创建索引。理想的做法是仅仅在常常被搜索的列（以及表）上面创建索引。

CREATE INDEX index\_name

ON table\_name (column\_name)

在表上创建一个唯一的索引。

使用UNIQUE INDEX （唯一的索引意味着两个行不能拥有相同的索引值）

撤销索引：DROP INDEX table\_name.index\_name

删除表：DROP TABLE table\_name

删除数据库：DROP DATABASE database\_name

ALTER TABLE 语句

用于在已有的表中添加、删除或修改列。

SQL AUTO INCREMENT 字段

在新记录插入表中时生成一个唯一的数字。

ID int NOT NULL AUTO\_INCREMENT

默认地，AUTO\_INCREMENT 的开始值是 1，每条新记录递增 1。

要让 AUTO\_INCREMENT 序列以其他的值起始，请使用下面的 SQL 语法：

ALTER TABLE Persons AUTO\_INCREMENT=100

SQL 视图（Views）

视图是可视化的表。

SQL CREATE VIEW 语法

CREATE VIEW view\_name AS

SELECT column\_name(s)

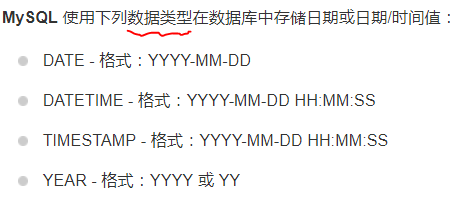
FROM table\_name

WHERE condition

SQL撤销视图：DROP VIEW view\_name

SQL日期函数（Date）





提示：如果您希望使查询简单且更易维护，那么请不要在日期中使用时间部分！

SQL NULL 值

如果表中的某个列是可选的，那么我们可以在不向该列添加值的情况下插入新记录或更新已有的记录。这意味着该字段将以 NULL 值保存。

NULL 值的处理方式与其他值不同。

NULL 用作未知的或不适用的值的占位符。

注意：NULL和0是不等价的

无法使用比较运算符来测试 NULL 值，比如 =、< 或 <>。

我们必须使用 IS NULL 和 IS NOT NULL 操作符。例如：

SELECT LastName,FirstName,Address FROM Persons

WHERE Address IS NULL

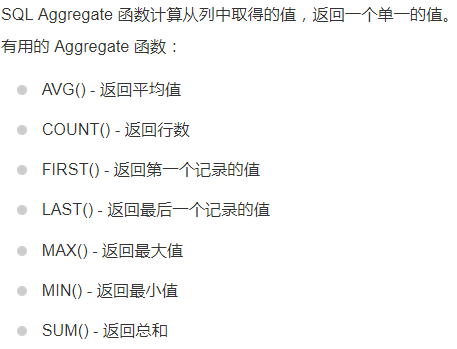
在 MySQL 中，我们可以使用 IFNULL() 函数，如下所示：

SELECT ProductName,UnitPrice\*(UnitsInStock+IFNULL(UnitsOnOrder,0))

FROM Products

此时NULL会返回0

SQL 拥有很多可用于计数和计算的内建函数。



COUNT(column\_name) 函数返回指定列的值的数目（NULL 不计入）

COUNT(\*) 函数返回表中的记录数

MySQL返回第一条记录

SELECT name FROM Websites

ORDER BY id ASC

LIMIT 1;

MySQL返回最后一条记录

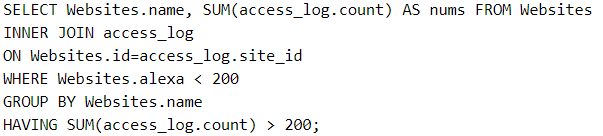
SELECT name FROM Websites

ORDER BY id DESC

LIMIT 1;

HAVING 子句

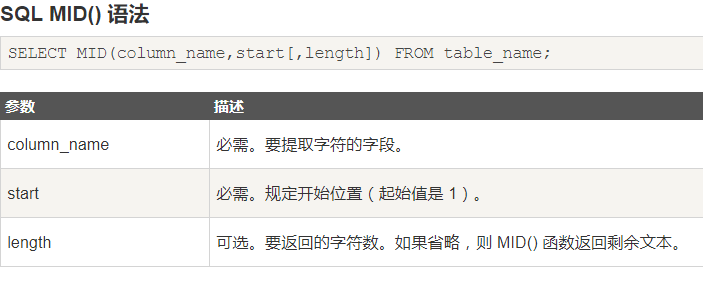
在 SQL 中增加 HAVING 子句原因是，WHERE 关键字无法与聚合函数一起使用，Group by可以，HAVING 子句可以让我们筛选分组后的各组数据。



查询排名小于200且访问量大于200的网站

MID() 函数

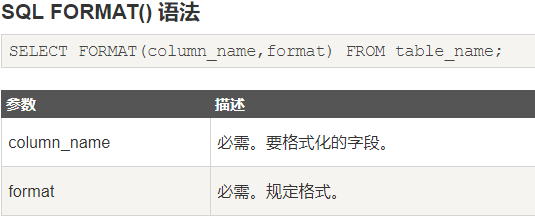
用于从文本字段中提取字符。



NOW() 函数，返回当前系统的日期和时间。

SELECT NOW() FROM table\_name;

FORMAT() 函数，用于对字段的显示进行格式化。





MySQL 使用标准的SQL数据语言形式。

网页是存储于名为 web 服务器的计算机中的文件。

读取网页的计算机可称为 web 客户机。

web 客户机通过名为 web 浏览器的程序来查看页面。

您的所有网页的集合被称为网站。

如果您的 PC 连入网络的话，您也可以把它当作一台 web 服务器。

大多数的情况是使用由 ISP（Internet 服务提供商）提供的 web 主机。

从一个 ISP 租用服务器是最常见的做法。这样做的好处有：

**连接速度**

大多数提供商都拥有极快的 Internet 连接。

**强大的硬件**

服务提供商通常拥有很多台可供多家公司分享的强大的 web 服务器。它们可提供负载平衡以及必要的备份服务。

**安全和稳定性**

ISP 是网站主机领域的专家。他们能够提供超过 99% 的正常服务时间、最新的软件漏洞补丁以及最好的病毒防护。

JSON：

JSON 指的是 JavaScript 对象表示法（JavaScript Object Notation）

JSON 是轻量级的文本数据交换格式

JSON 数组在中括号中书写：

数组可包含多个对象：

{

"sites": [

{ "name":"菜鸟教程" , "url":"www.runoob.com" },

{ "name":"google" , "url":"www.google.com" },

{ "name":"微博" , "url":"www.weibo.com" }

]

}

key 必须是字符串，value 可以是合法的 JSON 数据类型（字符串, 数字, 对象, 数组, 布尔值或 null）

key 和 value 中使用冒号分割。

每个 key/value 对使用逗号分割。

JSON 文件的文件类型是 ".json"

JSON 文本的 MIME 类型是 "application/json"

可以使用点号（.）来访问对象的值，也可以使用中括号（[]）来访问对象的值：

x = myObj.name; x = myObj["name"];

嵌套 JSON 对象

JSON 对象中可以包含另外一个 JSON 对象：

myObj = {

"name":"runoob",

"alexa":10000,

"sites": {

"site1":"www.runoob.com",

"site2":"m.runoob.com",

"site3":"c.runoob.com"

}

}