SQL：结构化查询语言，主流数据库系统都使用了SQL。

**功能：**

(1)、**建立**数据库的**表格**

(2)、改变数据库系统**环境设置**

(3)、针对某个数据库或表格，授予用户**存取权限**

(4)、对数据库表格**建立索引值**

(5)、修改数据库表格结构**(新建、删除或是修改表格字段)**

(6)、对数据库进行数据的**新建**

(7)、对数据库进行数据的**删除**

(8)、对数据库进行数据的**修改**

(9)、对数据库进行数据的**查询**

**SQL语言共分为四大类：**

数据查询语言**DQL**(以著名的SELECT语句为代表)；

数据操纵语言**DML**(INSERT、DELECT、UPDATE等语句)；

数据定义语言**DDL**(用来创建数据库对象——表、视图、同义词、索引等)；

数据控制语言**DCL**(用来授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，对数据库实行监控等)。

# 创建数据表

使用CREATE TABLE创建数据表，其语法为：

*CREATE TABLE tablename (field type [(size)] [NOT NULL]{, field type [(size)] [NOT NULL]...}[,PRIMARY KEY(field{,field...})]);*

**说明：**

table指定新建表的名称。

field指定在新表中创建的新字段的名称，每创建一个新表必须至少建立一个字段。

type指定新建字段的数据类型。

size指定文本及二进制字段的长度。

NOT NULL是SQL的关键字，使用该参数则新记录的该字段值必须是有效的数据。

PRIMARY KEY是规定哪个字段为主键。

**例**：

创建一个“HACK”的表，该表中有三个字段：

CREATE TABLE HACK(id char(4),name varchar(50),city varchar(50)，age char(150)，PRIMARY KEY (id));

有ID、name、city、age三个字段，其中ID为主键。在SQL中数据类型由13种基本类型组成，详见《sql中文参考手册》<http://www.shouce.ren/>

上面的例子中的中括号，大括号等一些符号的语法规则如下：

(1) 大写字母，在SQL语句中大小写是不区分的，我们在语法中把关键字规定为大写只是为了便于阅读。

(2) 小写字母为用户可指定的内容，例如CREATE的语法中的tablename，创建表时可以由我们自己定义的表名(例如HACK)所代替。

(3) 在语法描述中有一些特殊字符不能出现在用户书写的语句中。例如：{}、[]、|。

(4) []中的内容为可选的。例如语法中的[NOT NULL]中的NOT NULL可省略。

(5) {}表示若干个结构相似的内容的组合。例如{, field type [(size)] [NOT NULL]...}表示可定义多个列，列的定义之间用“，”分割。

(6) ｜表示在多个可选择的内容中选择其中一个。例如，如下语句：SELECT [ALL|DISTINCT] select\_list中的select\_list可用ALL或DISTINCT限定。

# 一、SQL数据查询

在SQL语言中，要实现对数据表中的数据进行查询功能，就要用到大名鼎鼎的SELECT语句了。这是一个功能强大使用灵活的语句，它的语法格式为：

SELECT[ALL|DISTINCT] {expr[[AS]c\_alias{,expr[[AS]c\_alias]...} }

FROM tableref{,tableref...}

[WHERE<search\_condition>]

expr为要查询的列名,多个列名之间使用“，”分隔。

tableref为查询的表。

search\_condition为查询的条件。

C\_alias为别名

简单查询

要查询表 HACK中所有黑客的信息:

SELECT \* FROM HACK

上述的查询语句省略了WHERE子句，因此它就实现了对HACK表中所有信息的查询。

在HACK表中查询城市为“长沙”的黑客的id和name。

SELECT id,name

FROM HACK

WHERE city= "长沙"

去掉查询结果中的重复纪录

在查询过程中，往往会常常出现重复的纪录，有的时候需要重复的纪录，而有的时候并不需要。SELECT语句已经提供了这个功能，如果要去掉查询结果中出现的相同元组纪录，可以使用DISTINCT关键字，例如SELECT DISTINCT id FROM HACK如果要保留查询结果中存在的相同元组，那么就需要使用ALL关键字，例如SELECT ALL id FROM HACK。

目标列中使用“\* ”符号

如查询语句SELECT \* FROM HACK。它的作用是查询HACK表中所有的信息。“\*”在SQL语句中代表所查询表中所有的列。所以上面的语句等于SELECT id,name,city,age FROM HACK。

在WHERE子句中使用逻辑运算符

要在HACK表中查询id<120且age>25的黑客信息：

SELECT \* FROM HACK WHERE id<120 AND age>25

在WHERE子句中使用逻辑运算符主要包括AND，OR，NOT。

AND运算符是对两个表达式进行逻辑“与”运算，reslut=expression1 AND expression2。其中reslut为结果，expression1和expression2为任意表达式。

AND的运算规则如表所示：

|  |  |  |
| --- | --- | --- |
| expression1的结果 | expression2的结果 | AND运算之后的结果 |
| 真 | 真 | 真 |
| 真 | 假 | 假 |
| 假 | 真 | 假 |
| 假 | 假 | 假 |

OR运算符是对两个表达式进行逻辑“或”运算，reslut=expression1 OR expression2。其中reslut为结果，expression1和expression2为任意表达式。

OR的运算规则如表所示：

|  |  |  |
| --- | --- | --- |
| expression1的结果 | expression2的结果 | OR运算之后的结果 |
| 真 | 真 | 真 |
| 真 | 假 | 真 |
| 假 | 真 | 真 |
| 假 | 假 | 假 |

NOT运算符是对表达式执行逻辑非运算，reslut=NOT expression。其中reslut为结果，expression为任意表达式。

NOT的运算规则如表所示：

|  |  |
| --- | --- |
| expression的结果 | NOT运算之后的结果 |
| 真 | 假 |
| 真 | 假 |

在WHERE子句中，使用的操作符主要有：<(小于)、<=（小于或等于）、>(大于)、>=（大于或等于）、=(等于)、<>(不等于)、Between(在某个取值范围内)、Like(匹配某个模式)、In(包含在某个值的列表中)。

下面详细说一下Between 和 Like 操作符，特别是在Like 操作符，它在搜索注入方面可是最重要的理论知识：

Between操作符返回的是位于界限之内的所有纪录值。比如查询HACK表中id字段1到100之间的全部纪录。SELECT \* FROM HACK WHERE id Between 1 AND 100

Like操作符所起的作用就是模糊查询，它在搜索方面运用的最多。目前很多系统并没有对搜索的地方进行过滤，所以可以利用它来进行注入。要使用Like进行模糊查询必须还要配合通配符的才能够完成。比如查询HACK表中city字段值中还有“京”字的所有黑客的信息：

SELECT \* FROM HACK WHERE city LIKE '%京%'

查询的结果就是city字段值中含有“京”的黑客的信息，比如可以查到北京的、南京的等等。不过需要注意的是在Like后的查询条件中要用单引号或双引号括起来。

Like操作符共有以下4种通配符，他们的含义如下：

%：代表零个或者多个任意字符

\_：代表一个任意字符

[]：指定范围内的任意单个字符

[^]：不在指定范围内的任意单个字符

下面举一些**例子**：

Like "BR%"返回以BR开始的任意字符串

Like "%een" 返回以een结束的任意字符串

Like "%en%" 返回包含en的任意字符串

Like"\_en" 返回以en结束的3个字符串

Like"[CK]%" 返回以C或者K开始的任意字符串

Like"[S-V]ing" 返回长为4个字符的字符串，结尾是"ing"，开始是从S到V

Like"M[^c]%" 返回以"M"开始且第二个字符不是"c"的任意字符串

在目标列中使用别名

在上面的HACK表中，所有的字段都是使用的英文，有的时候如果有看英文的习惯，那么就可以使用SQL中的一种别名的方法来给字段重新命名。使用AS关键字就可完成。例如把HACK表中的字段全部换成中文的。

SELECT id AS 标识, name AS 姓名, city AS 城市, age AS 年龄 FROM HACK

对查询结果进行排序

使用ORDER BY子句对查询结果按照一个或多个属性类进行排序。ORDER BY的语法是这样的：ORDER BY order\_by\_expression[ASC|DESC]{,order\_by\_expression[ASC|DESC]}其中，order\_by\_expression是要指定排序的列，可以将排序的列指定为列名、列的别名或表达式。ASC表示按照上升的循序排列，DESC表示按照降低的循序排列，默认的情况下是按照上升的循序排列。用HACK表来做例子，比如要查询city字段中含有“京”字的所有黑客信息，并按照id字段的降低循序排列：

SELECT \* FROM HACK WHERE city LIKE '%京%' ORDER BY id DESC

或者还可以这样查询city字段中含有“京”字的所有黑客信息，并按照age字段的降低循序排列，当age字段值相同时，按照id字段的上升排列：

SELECT \* FROM HACK WHERE city LIKE '%京%' ORDER BY age DESC, id ASC。

在SELECT中使用集合函数

为了进一步增强查询功能，SQL还提供了许多集合函数，主要有如下几种：

COUNT([DISTINCT|ALL] \* ) 统计元组的个数。DISTINCT关键字指明重复元组只统计一次，ALL指明重复元组需要重复统计。

COUNT([DISTINCT|ALL] column\_name ) 统计column\_name列非空的元组的数目。

SUM([DISTINCT|ALL] column\_name )计算column\_name列的值的总和，此列必须是数值型。

AVG([DISTINCT|ALL] column\_name )计算column\_name列的值的平均值，此列必须是数值型。

MAX([DISTINCT|ALL] column\_name )返回column\_name列中的最大值。

MIN([DISTINCT|ALL] column\_name )返回column\_name列中的最小值。

比如查询HACK表中年龄最大的、最小的和平均的年龄：

年龄最大的：SELECT MAX(age) FROM HACK

年龄最小的：SELECT MIN(age) FROM HACK

平均的年龄：SELECT AVG(age) FROM HACK

对查询结果进行分组

在查询语句中，还可以使用GROUP BY子句对查询结果按某一列或多列值进行分组。

语法为：GROUP BY [ALL] group\_by\_expression[,…..n]

例如查询HACK表中年龄相同的黑客的信息：

SELECT \* FROM HACK GROUP BY age

如果分组后还要按照某种条件对这些组来进行筛选，输出满足指定条件的元组的话，那么还可以使用HAVING子句指定筛选条件。

HAVING子句的语法为：HAVING search\_condition

其中search\_condition为指定要筛选的条件。使用HAVING子句需要注意三个方面：

**①** HAVING子句只能在GROUP BY子句后出现，不能够单独使用。

**②** HAVING子句只能对分组计算的结果进行筛选。

**③** HAVING子句不能够使用别名。

合并查询(Union Query)

合并查询(Union Query)是用于合并具有相同字段结构的两个表的内容，如果要在一个结果中集中显示不同表的内容，那么使用这个会非常有用。**例如**：

SELECT \* FROM HACK

UNION

SELECT \* FROM HACK1

这个查询结果把HACK和HACK1中的纪录合并到一个结果中，它的输出和远来表中的内容是一样的。在默认的情况下，合并查询不会返回重复纪录，不过可以在UNION后面加上关键字ALL而让合并查询显示重复纪录。

Union运算符允许把两个或者多个表的查询结果合并一个查询结果中。需要特别注意的是Union查询的表的字段必须是一样的，而且数据类型必须能够兼容。所以Union查询后的结果是增加了行数，而列数依然不变。这个Union查询在注入攻击的过程中非常重要。

连接查询

在实际的查询过程中，往往还需要同时从两个表或两个以上的表中查询数据。连接查询就允许同时从两个表或者两个以上表中查询数据，指定这些表中的某个或者某些列作为连接查询的条件。SQL中的连接查询包括等值连接、非等值连接、自然连接等。

连接查询的语法如下：

SELECT [ALL|DISTINCT] {expr[[AS]c\_alias[[AS]c\_alias]…. }]

FROM tableref,tableref

WHERE <join\_condition>

“join\_condition”为连接条件，它的一般格式如下：

[<表名 1>.]<列名 1> <比较运算符> [<表名 2>.]<列名 2>

其中比较运算符主要有：=，>，<，>=，<=，！=或<>

等值连接就是比较运算符为“=”的连接称为等值连接，而其他的就为非等值连接了。连接操作不仅可以在不同的表之间进行，也可以是一个表与它自身的连接。连接操作不仅可以在两个表中查询，而且可以是在两个或两个以上的连接，这被称为多表连接。

子查询

SELECT查询语句还可以支持嵌套查询，所谓嵌套就是指语句里面还可以含有相类似的语句。例如SELECT嵌套，就是指在SELECT查询语句中还可以含有SELECT语句。如果在一个SELECT语句中还嵌套了另外一个SELECT语句，那么被嵌套的SELECT语句就称为子查询。

子查询需要注意的有三个地方：

1. 子查询可以嵌套
2. 在子查询中不能使用ORDER BY子句，这个子句只对最后结果排序
3. 子查询要用括号括起来

例如查询HACK表中与id号为100的黑客在同一个城市的所有黑客的信息，代码如下：

SELECT \* FROM HACK

WHERE city=(SELECT city FROM HACK WHERE id=100)

当子查询返回的结果是一个集合时，可以使用IN、ANY/ALL或EXISTS这些叫做谓词的关键字进行判断。他们的含义分别如下：

[NOT]IN：元素与某一个集合进行比较，当这一个元素包含在这个集合中，那么结果为真，否则结果为假。NOT IN 的含义与 IN 相反。

ANY：含义为任何一个。例如“>ANY(子查询)”表示大于子查询结果中的某个值。

ALL：含义为所有。例如“>ALL(子查询)”表示大于子查询结果中的所有值。

[NOT]EXISTS：代表存在量词。例如“EXISTS(子查询)”当子查询没有返回纪录时结果为假，否则结果为真， NOT EXISTS的含义与EXISTS相反。

在脚本攻击中，与黑客紧密相连就要属EXISTS，例如：有两个表，一个为HACK，一个为CRACK。CRACK表中的字段和HACK中的字段是一样的。例如，要查询涉及HACK和CRACK表，可以在HACK表中依次取一个元组id，然后去查询CRACK表。如果CRACK表中存在这样的元组，它的id的值等于在HACK表中的值，那么在HACK表中的当前记录返回，代码如下：

SELECT \* FROM HACK

WHERE EXISTS

(SELECT \* FROM CRACK WHERE id=HACK.id)

使用EXISTS谓词，如果括号内的查询结果返回为空，那么EXISTS的结果为假，否则就为真。

# 二、SQL数据更新

INSERT用于向表中插入新的纪录，UPDATE用于修改表中某一纪录的值，DELETE用于删除中的纪录但不删除表的结构。

INSERT语句

INSERT语句用于向表中添加新的纪录，INSERT的语法如下：

INSERT INTO tablename [(colname{,colname…})]

{V ALUES (expr|NULL{,expr|NULL….})|Subquery}

向指定的表中插入新的纪录时，必须使用下面的两种方式的一种。

(1) 使用VALUES结构，这种情况下一次只能够向表中插入一条纪录。

(2) 使用子查询(Subquery)结构，这种方式可向表中插入多条纪录。即可在基于其他的表中的查询结构插入到当前的表中。

设置city为NULL可以有两种方式实现：

第一种：INSERT INTO HACK(id,name,age)

VALUES(101,曾云好,20)

这种情况下，插入的记录中的city当前值还不知道，所以在VALUES列表中没有给出这个字段的值而取他们的默认值(NULL)。

第二种：INSERT INTO HACK(id,name,age,city)

VALUES(101,曾云好,20,NULL)

这种情况下，明确地赋予city字段值为NULL。

将HACK表中city值为“长沙”的客户插入到CRACK这个表中。这个时候用到子查询(Subquery)结构，其代码为：

INSERT INTO CRACK

SELECT \* FROM HACK WHERE city="长沙"

不过基于子查询(Subquery)结构中需要注意的是，

一、基于其他表查询的INSERT INTO语句不需要在目标表中指出列名

二、INSERT INTO目标表中的列名数目和顺序必须和SELECT目标列中的数目和顺序一致

UPDATE语句

UPDATE语句用来修改表中纪录的属性值，它的基本语法格式如下：

UPDATE tablename

SET colname={expr|NULL|(subquery)}

{, colname={expr|NULL|(subquery)…..}}

[WH ERE search\_condition]

UPDATE语句使用指定的表达式的值来代替所满足WHERE条件的纪录中指定列的值。UPDATE语句同样支持子查询(Subquery)结构。把HACK表中city值为“长沙”的黑客的年龄全部提高10岁，语句如下：

UPDATE HACK

SET age=10+age

WHERE city="长沙"

DELETE语句

DELETE语句用于从表中删除满足条件的纪录，DELETE语句的基本语法格式如下：

DELETE FROM tablename

[WHERE search\_condition]

删除city为“长沙”的所有黑客的信息：

DELETE FROM HACK

WHERE city="长沙"

删除HACK表中的所有信息：

DELETE FROM HACK

不过需要注意的是DELETE只删除HACK表中的所有纪录，并不删除表的结构，也就是说表还是存在，只是里面的数据没了而已。