# 一、SQL简介

## 1、什么是 SQL？

**SQL 是用于访问和处理数据库的标准的计算机语言。**

* SQL 指结构化查询语言
* SQL 使我们有能力访问数据库
* SQL 是一种 ANSI 的标准计算机语言

**编者注：**ANSI，美国国家标准化组织

## 2、SQL 能做什么？

* SQL 面向数据库执行查询
* SQL 可从数据库取回数据
* SQL 可在数据库中插入新的记录
* SQL 可更新数据库中的数据
* SQL 可从数据库删除记录
* SQL 可创建新数据库
* SQL 可在数据库中创建新表
* SQL 可在数据库中创建存储过程
* SQL 可在数据库中创建视图
* SQL 可以设置表、存储过程和视图的权限

## 3、SQL 是一种标准

SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MS Access、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。

不幸地是，存在着很多不同版本的 SQL 语言，但是为了与 ANSI 标准相兼容，它们必须以相似的方式共同地来支持一些主要的关键词（比如 SELECT、UPDATE、DELETE、INSERT、WHERE 等等）。

**注释：**除了 SQL 标准之外，大部分 SQL 数据库程序都拥有它们自己的私有扩展！

## 4、RDBMS

RDBMS 指的是关系型数据库管理系统。

RDBMS 是 SQL 的基础，同样也是所有现代数据库系统的基础，比如 MS SQL Server, IBM DB2, Oracle, MySQL 以及 Microsoft Access。

RDBMS 中的数据存储在被称为表（tables）的数据库对象中。

表是相关的数据项的集合，它由列和行组成。

# 二、SQL语法

## 1、数据库表

一个数据库通常包含一个或多个表。每个表由一个名字标识（例如“客户”或者“订单”）。表包含带有数据的记录（行）。

下面的例子是一个名为 "Persons" 的表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

上面的表包含三条记录（每一条对应一个人）和五个列（Id、姓、名、地址和城市）。

**2、SQL 语句**

您需要在数据库上执行的大部分工作都由 SQL 语句完成。

下面的语句从表中选取LastName列的数据：

SELECT LastName FROM Persons

结果集类似这样：

|  |
| --- |
| **LastName** |
| Adams |
| Bush |
| Carter |

**3、SQL 语句后面的分号？**

某些数据库系统要求在每条 SQL 命令的末端使用分号。

分号是在数据库系统中分隔每条 SQL 语句的标准方法，这样就可以在对服务器的相同请求中执行一条以上的语句。

**4、DML 和 DDL**

可以把 SQL 分为两个部分：数据操作语言 (DML) 和数据定义语言 (DDL)。

SQL (结构化查询语言)是用于执行查询的语法。但是 SQL 语言也包含用于更新、插入和删除记录的语法。

查询和更新指令构成了 SQL 的 DML 部分：

* SELECT - 从数据库表中获取数据
* UPDATE - 更新数据库表中的数据
* DELETE - 从数据库表中删除数据
* INSERT INTO - 向数据库表中插入数据

SQL 的数据定义语言 (DDL) 部分使我们有能力创建或删除表格。我们也可以定义索引（键），规定表之间的链接，以及施加表间的约束。

SQL 中最重要的 DDL 语句:

* 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

# 三、SELECT 语句

SELECT 语句用于从表中选取数据。

结果被存储在一个结果表中（称为结果集）。

## 1、SELECT 语法

SELECT 列名称 FROM 表名称

以及：

SELECT \* FROM 表名称

**注释：**SQL 语句对大小写不敏感。SELECT 等效于 select。

## 2、SELECT 实例

如需获取名为 "LastName" 和 "FirstName" 的列的内容（从名为 "Persons" 的数据库表），请使用类似这样的 SELECT 语句：

SELECT LastName,FirstName FROM Persons

**"Persons" 表:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

**结果：**

|  |  |
| --- | --- |
| **LastName** | **FirstName** |
| Adams | John |
| Bush | George |
| Carter | Thomas |

## 3、SELECT \* 实例

现在我们希望从 "Persons" 表中选取所有的列。

请使用符号 \* 取代列的名称，就像这样：

SELECT \* FROM Persons

**提示：**星号（\*）是选取所有列的快捷方式。

**结果：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

# 四、WHERE 子句

如需有条件地从表中选取数据，可将 WHERE 子句添加到 SELECT 语句。

## 1、语法

SELECT 列名称 FROM 表名称 WHERE 列运算符值

下面的运算符可在 WHERE 子句中使用：

|  |  |
| --- | --- |
| **操作符** | **描述** |
| = | 等于 |
| <> | 不等于 |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| BETWEEN AND | 在某个范围内 |
| LIKE | 搜索某种模式 |

**注释：**在某些版本的 SQL 中，操作符<>可以写为 !=。

**2、使用 WHERE 子句**

如果只希望选取居住在城市 "Beijing" 中的人，我们需要向 SELECT 语句添加 WHERE 子句：

SELECT \* FROM Persons WHERE City='Beijing'

**"Persons" 表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** | **Year** |
| Adams | John | Oxford Street | London | 1970 |
| Bush | George | Fifth Avenue | New York | 1975 |
| Carter | Thomas | Changan Street | Beijing | 1980 |
| Gates | Bill | Xuanwumen 10 | Beijing | 1985 |

**结果：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** | **Year** |
| Carter | Thomas | Changan Street | Beijing | 1980 |
| Gates | Bill | Xuanwumen 10 | Beijing | 1985 |

**3、引号的使用**

请注意，我们在例子中的条件值周围使用的是单引号。

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**，请不要使用引号。

**文本值：**

这是正确的：

SELECT \* FROM Persons WHERE FirstName=“Bush”

这是错误的：

SELECT \* FROM Persons WHERE FirstName=Bush

**数值：**

这是正确的：

SELECT \* FROM Persons WHERE Year>1965

这是错误的：

SELECT \* FROM Persons WHERE Year>'1965'

# 五、AND 和 OR 运算符

AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

如果第一个条件和第二个条件都成立，则 AND 运算符显示一条记录。

如果第一个条件和第二个条件中只要有一个成立，则 OR 运算符显示一条记录。

**原始表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Adams | John | Oxford Street | London |
| Bush | George | Fifth Avenue | New York |
| Carter | Thomas | Changan Street | Beijing |
| Carter | William | Xuanwumen 10 | Beijing |

## 1、AND 运算符实例

使用 AND 来显示所有姓为 "Carter" 并且名为 "Thomas" 的人：

SELECT \* FROM Persons WHERE FirstName='Thomas' AND LastName='Carter'

**结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |

## 2、OR 运算符实例

使用 OR 来显示所有姓为 "Carter" 或者名为 "Thomas" 的人：

SELECT \* FROM Persons WHERE firstname='Thomas' OR lastname='Carter'

**结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |
| Carter | William | Xuanwumen 10 | Beijing |

**3、结合 AND 和 OR 运算符**

我们也可以把 AND 和 OR 结合起来（使用圆括号来组成复杂的表达式）:

SELECT \* FROM Persons WHERE (FirstName='Thomas' OR FirstName='William') AND LastName='Carter'

**结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |
| Carter | William | Xuanwumen 10 | Beijing |

**ORDER BY 语句用于对结果集进行排序。**

# 六、ORDER BY 语句

ORDER BY 语句用于根据指定的列对结果集进行排序。

ORDER BY 语句默认按照升序对记录进行排序。

如果您希望按照降序对记录进行排序，可以使用 DESC 关键字。

**原始表**

Orders 表:

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| IBM | 3532 |
| Hwadee | 2356 |
| Apple | 4698 |
| Hwadee | 6953 |

**1、实例 1**

以字母顺序显示公司名称：

SELECT Company, OrderNumber FROM Orders ORDER BY Company

**结果：**

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| Apple | 4698 | |
| IBM | 3532 | |
| Hwadee | 6953 | |
| Hwadee | 2356 | |

**2、实例 2**

以字母顺序显示公司名称（Company），并以数字顺序显示顺序号（OrderNumber）：

SELECT Company, OrderNumber FROM Orders ORDER BY Company, OrderNumber

结果：

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| Apple | 4698 |
| IBM | 3532 |
| Hwadee | 2356 |
| Hwadee | 6953 |

**3、实例 3**

以逆字母顺序显示公司名称：

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC

**结果：**

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| Hwadee | 6953 |
| Hwadee | 2356 |
| IBM | 3532 |
| Apple | 4698 |

**4、实例 4**

以逆字母顺序显示公司名称，并以数字顺序显示顺序号：

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC, OrderNumber ASC

**结果：**

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| Hwadee | 2356 |
| Hwadee | 6953 |
| IBM | 3532 |
| Apple | 4698 |

**注意：**在以上的结果中有两个相等的公司名称 (Hwadee)。只有这一次，在第一列中有相同的值时，第二列是以升序排列的。如果第一列中有些值为 nulls 时，情况也是这样的。

# 七、SELECT DISTINCT 语句

在表中，可能会包含重复值。这并不成问题，不过，有时您也许希望仅仅列出不同（distinct）的值。

关键词 DISTINCT 用于返回唯一不同的值。

## 1、语法：

SELECT DISTINCT 列名称 FROM 表名称

## 2、使用 DISTINCT 关键词

如果要从 "Company" 列中选取所有的值，我们需要使用 SELECT 语句：

SELECT Company FROM Orders

**"Orders"表：**

|  |  |
| --- | --- |
| **Company** | **OrderNumber** |
| IBM | 3532 |
| Hwadee | 2356 |
| Apple | 4698 |
| Hwadee | 6953 |

**结果：**

|  |
| --- |
| **Company** |
| IBM |
| Hwadee |
| Apple |
| Hwadee |

请注意，在结果集中，Hwadee 被列出了两次。

如需从 Company" 列中仅选取唯一不同的值，我们需要使用 SELECT DISTINCT 语句：

SELECT DISTINCT Company FROM Orders

**结果：**

|  |
| --- |
| **Company** |
| IBM |
| Hwadee |
| Apple |

现在，在结果集中，"Hwadee" 仅被列出了一次。

# 八、INSERT INTO 语句

INSERT INTO 语句用于向表格中插入新的行。

## 1、语法

INSERT INTO 表名称 VALUES (值1, 值2,....)

我们也可以指定所要插入数据的列：

INSERT INTO table\_name (列1, 列2,...) VALUES (值1, 值2,....)

## 2、插入新的行

**"Persons" 表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |

**SQL 语句：**

INSERT INTO Persons VALUES ('Gates', 'Bill', 'Xuanwumen 10', 'Beijing')

**结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |

## 3、在指定的列中插入数据

**"Persons" 表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |

**SQL 语句：**

INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees')

**结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Carter | Thomas | Changan Street | Beijing |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson |  | Champs-Elysees |  |

# 九、Update 语句

Update 语句用于修改表中的数据。

## 1、语法：

UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

**Person:**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson |  | Champs-Elysees |  |

## 2、更新某一行中的一个列

我们为lastname是 "Wilson" 的人添加firstname：

UPDATE Person SET FirstName = 'Fred' WHERE LastName = 'Wilson'

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Champs-Elysees |  |

## 3、更新某一行中的若干列

我们会修改地址（address），并添加城市名称（city）：

UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing'

WHERE LastName = 'Wilson'

结果：

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Zhongshan 23 | Nanjing |

# 十、DELETE 语句

DELETE 语句用于删除表中的行。

## 1、语法

DELETE FROM 表名称 WHERE 列名称 = 值

**Person表:**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Gates | Bill | Xuanwumen 10 | Beijing |
| Wilson | Fred | Zhongshan 23 | Nanjing |

## 2、删除某行

"Fred Wilson" 会被删除：

DELETE FROM Person WHERE LastName = 'Wilson'

**结果:**

|  |  |  |  |
| --- | --- | --- | --- |
| **LastName** | **FirstName** | **Address** | **City** |
| Gates | Bill | Xuanwumen 10 | Beijing |

## 3、删除所有行

可以在不删除表的情况下删除所有的行。这意味着表的结构、属性和索引都是完整的：

DELETE FROM table\_name

或者：

DELETE \* FROM table\_name

# 十一、Alias

通过使用 SQL，可以为列名称和表名称指定别名（Alias）。

## 1、表的 Alias 语法

SELECT column\_name(s)

FROM table\_name

AS alias\_name

## 2、列的 Alias 语法

SELECT column\_name AS alias\_name

FROM table\_name

## 3、Alias 实例: 使用表名称别名

假设我们有两个表分别是："Persons" 和 "Product\_Orders"。我们分别为它们指定别名 "p" 和 "po"。

现在，我们希望列出 "John Adams" 的所有定单。

我们可以使用下面的 SELECT 语句：

SELECT po.OrderID, p.LastName, p.FirstName

FROM Persons AS p, Product\_Orders AS po

WHERE p.LastName='Adams' AND p.FirstName='John'

**不使用别名的 SELECT 语句：**

SELECT Product\_Orders.OrderID, Persons.LastName, Persons.FirstName

FROM Persons, Product\_Orders

WHERE Persons.LastName='Adams' AND Persons.FirstName='John'

从上面两条 SELECT 语句您可以看到，别名使查询程序更易阅读和书写。

## 4、Alias 实例: 使用一个列名别名

**表 Persons:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

**SQL:**

SELECT LastName AS Family, FirstName AS Name

FROM Persons

**结果：**

|  |  |
| --- | --- |
| **Family** | **Name** |
| Adams | John |
| Bush | George |
| Carter | Thomas |

**在搜索数据库中的数据时，您可以使用 SQL 通配符。**

# 十二.通配符

在搜索数据库中的数据时，SQL 通配符可以替代一个或多个字符。

SQL 通配符必须与 LIKE 运算符一起使用。

在 SQL 中，可使用以下通配符：

|  |  |
| --- | --- |
| **通配符** | **描述** |
| % | 替代一个或多个字符 |
| \_ | 仅替代一个字符 |

**原始的表 (用在例子中的)：**

Persons 表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

## 1、使用 % 通配符

**例子 1**

现在，我们希望从上面的 "Persons" 表中选取居住在以 "Ne" 开始的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons WHERE City LIKE '%Ne%'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Bush | George | Fifth Avenue | New York |

**例子 2**

接下来，我们希望从 "Persons" 表中选取居住在包含 "lond" 的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE City LIKE '%lond%'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |

## 2、使用 \_ 通配符

**例子 1**

现在，我们希望从上面的 "Persons" 表中选取名字的第一个字符之后是 "eorge" 的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons WHERE FirstName LIKE '\_eorge'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Bush | George | Fifth Avenue | New York |

**例子 2**

接下来，我们希望从 "Persons" 表中选取的这条记录的姓氏以 "C" 开头，然后是一个任意字符，然后是 "r"，然后是任意字符，然后是 "er"：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE LastName LIKE 'C\_r\_er'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 3 | Carter | Thomas | Changan Street | Beijing |

# 十三、LIMIT 子句

limit 子句用于规定要返回的记录的数目。

对于拥有数千条记录的大型表来说，limit子句是非常有用的。

**注释：**并非所有的数据库系统都支持 limit子句。

**MySQL 语法**

SELECT column\_name(s)

FROM table\_name

LIMIT number

**例子**

SELECT \*

FROM Persons

LIMIT 5

**原始的表 (用在例子中的)：**

Persons 表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |
| 4 | Obama | Barack | Pennsylvania Avenue | Washington |

# 十四、LIKE 操作符

LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式。

## 1、LIKE 操作符语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name LIKE pattern

**原始的表 (用在例子中的)：**

Persons 表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

## 2、LIKE 操作符实例

**例子 1**

现在，我们希望从上面的 "Persons" 表中选取居住在以 "N" 开始的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE City LIKE 'N%'

**提示：**"%" 可用于定义通配符（模式中缺少的字母）。

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Bush | George | Fifth Avenue | New York |

**例子 2**

接下来，我们希望从 "Persons" 表中选取居住在以 "g" 结尾的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons WHERE City LIKE '%g'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 3 | Carter | Thomas | Changan Street | Beijing |

**例子 3**

接下来，我们希望从 "Persons" 表中选取居住在包含 "lon" 的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE City LIKE '%lon%'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |

**例子 4**

通过使用 NOT 关键字，我们可以从 "Persons" 表中选取居住在**不包含** "lon" 的城市里的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE City NOT LIKE '%lon%'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

# 十五、IN 操作符

IN 操作符允许我们在 WHERE 子句中规定多个值。

## 1、IN 语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name IN (value1,value2,...)

**原始的表 (在实例中使用：)**

Persons 表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |

## 2、IN 操作符实例

现在，我们希望从上表中选取姓氏为 Adams 和 Carter 的人：

我们可以使用下面的 SELECT 语句：

SELECT \* FROM Persons

WHERE LastName IN ('Adams','Carter')

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 3 | Carter | Thomas | Changan Street | Beijing |

# 十六、BETWEEN 操作符

操作符 BETWEEN ... AND 会选取介于两个值之间的数据范围。这些值可以是数值、文本或者日期。

Persons 表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |
| 3 | Carter | Thomas | Changan Street | Beijing |
| 4 | Gates | Bill | Xuanwumen 10 | Beijing |

## 1、实例 1：

如需以字母顺序显示介于 "Adams"（包括）和 "Carter"（不包括）之间的人，请使用下面的 SQL：

SELECT \* FROM Persons

WHERE LastName

BETWEEN 'Adams' AND 'Carter'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Adams | John | Oxford Street | London |
| 2 | Bush | George | Fifth Avenue | New York |

**重要事项：**不同的数据库对 BETWEEN...AND 操作符的处理方式是有差异的。某些数据库会列出介于 "Adams" 和 "Carter" 之间的人，但不包括 "Adams" 和 "Carter" ；某些数据库会列出介于 "Adams" 和 "Carter" 之间并包括 "Adams" 和 "Carter" 的人；而另一些数据库会列出介于 "Adams" 和 "Carter" 之间的人，包括 "Adams" ，但不包括 "Carter" 。

## 2、实例 2：

如需使用上面的例子显示范围之外的人，请使用 NOT 操作符：

SELECT \* FROM Persons

WHERE LastName

NOT BETWEEN 'Adams' AND 'Carter'

**结果集：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **LastName** | **FirstName** | **Address** | **City** |
| 3 | Carter | Thomas | Changan Street | Beijing |
| 4 | Gates | Bill | Xuanwumen 10 | Beijing |