**SQL 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。

**语法：**

SELECT DISTINCT 列名称 FROM 表名称

## 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。

## 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。

SELECT Company, OrderNumber FROM Orders ORDER BY Company

## INSERT INTO 语句

INSERT INTO 语句用于向表格中插入新的行。

### 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,....)

$sql="insert into motto(id,writer,content,time) values(2,'冯上桂','asdasda',1514151)";

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

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

## DELETE FROM 表名称 WHERE 列名称 = 值

## DELETE FROM Person WHERE LastName = 'Wilson' 删除某行

## DELETE \* FROM table\_name删除所有行

# SQL TOP 子句

TOP 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，TOP 子句是非常有用的。

**注释：**并非所有的数据库系统都支持 TOP 子句。

### SQL Server 的语法：

SELECT TOP number|percent column\_name(s)

FROM table\_name

SELECT TOP 2 \* FROM Persons

SELECT TOP 50 PERCENT \* FROM Persons

## 现在，我们希望从上面的 "Persons" 表中选取头两条记录。

## 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 操作符语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name LIKE pattern

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）。

|  |  |
| --- | --- |
| **通配符** | **描述** |
| % | 代表零个或多个字符 |
| \_ （下划线） | 仅替代一个字符 |
| [charlist] | 字符列中的任何单一字符 |
| [^charlist]  或者  [!charlist] | 不在字符列中的任何单一字符 |

## IN 操作符

IN 操作符允许我们在 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,...)

## 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')

## 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。

### SQL BETWEEN 语法

SELECT column\_name(s)

FROM table\_name

WHERE column\_name

BETWEEN value1 AND value2

## BETWEEN 操作符实例

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'

如需使用上面的例子显示范围之外的人，请使用 NOT 操作符： 在betwwen前加not

**通过使用 SQL，可以为列名称和表名称指定别名（Alias）。**

## SQL Alias

### 表的 SQL Alias 语法

SELECT column\_name(s)

FROM table\_name

AS alias\_name

### 列的 SQL Alias 语法

SELECT column\_name AS alias\_name

FROM table\_name

## Alias 实例: 使用一个列名别名

SELECT LastName AS Family, FirstName AS Name

FROM Persons

## **SQL join 用于根据两个或多个表中的列之间的关系，从这些表中查询数据。**引用两个表

我们可以通过引用两个表的方式，从两个表中获取数据：

谁订购了产品，并且他们订购了什么产品？

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons, Orders

WHERE Persons.Id\_P = Orders.Id\_P

## SQL JOIN - 使用 Join

除了上面的方法，我们也可以使用关键词 JOIN 来从两个表中获取数据。

如果我们希望列出所有人的定购，可以使用下面的 SELECT 语句：

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

INNER JOIN Orders

ON Persons.Id\_P = Orders.Id\_P

ORDER BY Persons.LastName

## SQL INNER JOIN 关键字

在表中存在至少一个匹配时，INNER JOIN 关键字返回行。

SELECT column\_name(s)

FROM table\_name1

INNER JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

电脑萤幕的截图

描述已自动生成

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

INNER JOIN Orders语法

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

图形用户界面

中度可信度描述已自动生成

## SQL LEFT JOIN 关键字

LEFT JOIN 关键字会从左表 (table\_name1) 那里返回所有的行，即使在右表 (table\_name2) 中没有匹配的行。

### LEFT JOIN 关键字语法

SELECT column\_name(s)

FROM table\_name1

LEFT JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

**注释：**在某些数据库中， LEFT JOIN 称为 LEFT OUTER JOIN。

## 左连接（LEFT JOIN）实例

现在，我们希望列出所有的人，以及他们的定购 - 如果有的话。

您可以使用下面的 SELECT 语句：

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

LEFT JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

表格

描述已自动生成

**右连接一样**

## SQL FULL JOIN 关键字

只要其中某个表存在匹配，FULL JOIN 关键字就会返回行。

### FULL JOIN 关键字语法

SELECT column\_name(s)

FROM table\_name1

FULL JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

**注释：**在某些数据库中， FULL JOIN 称为 FULL OUTER JOIN。

## 全连接（FULL JOIN）实例

现在，我们希望列出所有的人，以及他们的定单，以及所有的定单，以及定购它们的人。

您可以使用下面的 SELECT 语句：

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

FULL JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

## SQL UNION 操作符

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。

请注意，UNION 内部的 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。同时，每条 SELECT 语句中的列的顺序必须相同。

### SQL UNION 语法

SELECT column\_name(s) FROM table\_name1

UNION

SELECT column\_name(s) FROM table\_name2

**注释：**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 UNION ALL 语法

SELECT column\_name(s) FROM table\_name1

UNION ALL

SELECT column\_name(s) FROM table\_name2

另外，UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。

列出所有在中国和美国的不同的雇员名：

SELECT E\_Name FROM Employees\_China

UNION

SELECT E\_Name FROM Employees\_USA

**注释：**这个命令无法列出在中国和美国的所有雇员。在上面的例子中，我们有两个名字相同的雇员，他们当中只有一个人被列出来了。UNION 命令只会选取不同的值。 意思就是说如果有两个人同名 只选一个。

## 使用 UNION ALL 命令

### 实例：

列出在中国和美国的所有的雇员：

SELECT E\_Name FROM Employees\_China

UNION ALL

SELECT E\_Name FROM Employees\_USA

这个就是可以全部都包括在内。

## SELECT INTO 语句

SELECT INTO 语句从一个表中选取数据，然后把数据插入另一个表中。

SELECT INTO 语句常用于创建表的备份复件或者用于对记录进行存档。

### SQL SELECT INTO 语法

您可以把所有的列插入新表：

SELECT \*

INTO new\_table\_name [IN externaldatabase]

FROM old\_tablename

或者只把希望的列插入新表：

SELECT column\_name(s)

INTO new\_table\_name [IN externaldatabase]

FROM old\_tablename

文本

描述已自动生成

## 图形用户界面, 文本, 应用程序, 电子邮件 描述已自动生成CREATE DATABASE 语句

CREATE DATABASE 用于创建数据库。

### SQL CREATE DATABASE 语法

CREATE DATABASE database\_name

### SQL CREATE TABLE 语法

CREATE TABLE 表名称

(

列名称1 数据类型,

列名称2 数据类型,

列名称3 数据类型,

....

)

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| * integer(size) * int(size) * smallint(size) * tinyint(size) | 仅容纳整数。在括号内规定数字的最大位数。 |
| * decimal(size,d) * numeric(size,d) | 容纳带有小数的数字。  "size" 规定数字的最大位数。"d" 规定小数点右侧的最大位数。 |
| char(size) | 容纳固定长度的字符串（可容纳字母、数字以及特殊字符）。  在括号中规定字符串的长度。 |
| varchar(size) | 容纳可变长度的字符串（可容纳字母、数字以及特殊的字符）。  在括号中规定字符串的最大长度。 |
| date(yyyymmdd) | 容纳日期。 |

该表包含 5 个列，列名分别是："Id\_P"、"LastName"、"FirstName"、"Address" 以及 "City"：

CREATE TABLE Persons

(

Id\_P int,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

**SQL 约束**

约束用于限制加入表的数据的类型。

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

我们将主要探讨以下几种约束：

* NOT NULL
* UNIQUE
* PRIMARY KEY
* FOREIGN KEY
* CHECK
* DEFAULT

## SQL NOT NULL 约束

NOT NULL 约束强制列不接受 NULL 值。

NOT NULL 约束强制字段始终包含值。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。

下面的 SQL 语句强制 "Id\_P" 列和 "LastName" 列不接受 NULL 值：

Id\_P int NOT NULL,

## SQL UNIQUE 约束

UNIQUE 约束唯一标识数据库表中的每条记录。

UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。

PRIMARY KEY 拥有自动定义的 UNIQUE 约束。

请注意，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。

### SQL Server / Oracle / MS Access:

CREATE TABLE Persons

(//MYSQL语法不同 这个unique

Id\_P int NOT NULL UNIQUE,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

### 如果需要命名 UNIQUE 约束，以及为多个列定义 UNIQUE 约束，请使用下面的 SQL 语法： 全部都通

CONSTRAINT uc\_PersonID UNIQUE (Id\_P,LastName)

### MySQL / SQL Server / Oracle / MS Access:

当表已被创建时，如需在 "Id\_P" 列创建 UNIQUE 约束，请使用下列 SQL：

ALTER TABLE Persons

ADD UNIQUE (Id\_P)

如需命名 UNIQUE 约束，并定义多个列的 UNIQUE 约束，请使用下面的 SQL 语法：

### MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ADD CONSTRAINT uc\_PersonID UNIQUE (Id\_P,LastName)

## 撤销 UNIQUE 约束

如需撤销 UNIQUE 约束，请使用下面的 SQL：

### SQL Server / Oracle / MS Access:

ALTER TABLE Persons

DROP CONSTRAINT uc\_PersonID

## SQL PRIMARY KEY 约束

## references是在为表创建外键时的一个固定语法里的词。

## 电脑萤幕的截图 描述已自动生成

PRIMARY KEY 约束唯一标识数据库表中的每条记录。

主键必须包含唯一的值。

主键列不能包含 NULL 值。

每个表都应该有一个主键，并且每个表只能有一个主键。

### 语法和上述所讲都对应，包括通用和不通用

## SQL FOREIGN KEY 约束

一个表中的 FOREIGN KEY 指向另一个表中的 PRIMARY KEY。

下面的 SQL 在 "Orders" 表创建时为 "Id\_P" 列创建 FOREIGN KEY：

### SQL Server / Oracle / MS Access:

CREATE TABLE Orders

(

Id\_O int NOT NULL PRIMARY KEY,

OrderNo int NOT NULL,

Id\_P int FOREIGN KEY REFERENCES Persons(Id\_P)

)

如果需要命名 FOREIGN KEY 约束，以及为多个列定义 FOREIGN KEY 约束，请使用下面的 SQL 语法：

### MySQL / SQL Server / Oracle / MS Access:

CREATE TABLE Orders

(

Id\_O int NOT NULL,

OrderNo int NOT NULL,

Id\_P int,

PRIMARY KEY (Id\_O),

CONSTRAINT fk\_PerOrders FOREIGN KEY (Id\_P)

REFERENCES Persons(Id\_P) //第一个表

)

如果在 "Orders" 表已存在的情况下为 "Id\_P" 列创建 FOREIGN KEY 约束，请使用下面的 SQL：

### MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Orders

ADD FOREIGN KEY (Id\_P)

REFERENCES Persons(Id\_P)

如果需要命名 FOREIGN KEY 约束，以及为多个列定义 FOREIGN KEY 约束，请使用下面的 SQL 语法：

### MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Orders

ADD CONSTRAINT fk\_PerOrders

FOREIGN KEY (Id\_P)

REFERENCES Persons(Id\_P)

## 撤销 FOREIGN KEY 约束

如需撤销 FOREIGN KEY 约束，请使用下面的 SQL：

### SQL Server / Oracle / MS Access:

ALTER TABLE Orders

DROP CONSTRAINT fk\_PerOrders

## SQL CHECK 约束

CHECK 约束用于限制列中的值的范围。

如果对单个列定义 CHECK 约束，那么该列只允许特定的值。

如果对一个表定义 CHECK 约束，那么此约束会在特定的列中对值进行限制。

下面的 SQL 在 "Persons" 表创建时为 "Id\_P" 列创建 CHECK 约束。CHECK 约束规定 "Id\_P" 列必须只包含大于 0 的整数。

### SQL Server / Oracle / MS Access:

CREATE TABLE Persons

(

Id\_P int NOT NULL CHECK (Id\_P>0),

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

如果需要命名 CHECK 约束，以及为多个列定义 CHECK 约束，请使用下面的 SQL 语法：

### MySQL / SQL Server / Oracle / MS Access:

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT chk\_Person CHECK (Id\_P>0 AND City='Sandnes')

)

如果在表已存在的情况下为 "Id\_P" 列创建 CHECK 约束，请使用下面的 SQL：

### MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ADD CHECK (Id\_P>0)

如果需要命名 CHECK 约束，以及为多个列定义 CHECK 约束，请使用下面的 SQL 语法：

### MySQL / SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ADD CONSTRAINT chk\_Person CHECK (Id\_P>0 AND City='Sandnes')

## 撤销 CHECK 约束

如需撤销 CHECK 约束，请使用下面的 SQL：

### SQL Server / Oracle / MS Access:

ALTER TABLE Persons

DROP CONSTRAINT chk\_Person

## SQL DEFAULT 约束

DEFAULT 约束用于向列中插入默认值。

如果没有规定其他的值，那么会将默认值添加到所有的新记录。

下面的 SQL 在 "Persons" 表创建时为 "City" 列创建 DEFAULT 约束：

### My SQL / SQL Server / Oracle / MS Access:

City varchar(255) DEFAULT 'Sandnes'

)

### 如果在表已存在的情况下为 "City" 列创建 DEFAULT 约束，请使用下面的 SQL：

### SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ALTER COLUMN City SET DEFAULT 'SANDNES'

## 撤销 DEFAULT 约束

### SQL Server / Oracle / MS Access:

ALTER TABLE Persons

ALTER COLUMN City DROP DEFAULT

**CREATE INDEX 语句用于在表中创建索引。**

**在不读取整个表的情况下，索引使数据库应用程序可以更快地查找数据。**

## 索引

您可以在表中创建索引，以便更加快速高效地查询数据。

用户无法看到索引，它们只能被用来加速搜索/查询。

**注释：**更新一个包含索引的表需要比更新一个没有索引的表更多的时间，这是由于索引本身也需要更新。因此，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。

### SQL CREATE INDEX 语法

在表上创建一个简单的索引。允许使用重复的值：

CREATE INDEX index\_name

ON table\_name (column\_name)

**注释：**"column\_name" 规定需要索引的列。

### SQL CREATE UNIQUE INDEX 语法

在表上创建一个唯一的索引。唯一的索引意味着两个行不能拥有相同的索引值。

CREATE UNIQUE INDEX index\_name

ON table\_name (column\_name)

## CREATE INDEX 实例

本例会创建一个简单的索引，名为 "PersonIndex"，在 Person 表的 LastName 列：

CREATE INDEX PersonIndex

ON Person (LastName)

如果您希望以**降序**索引某个列中的值，您可以在列名称之后添加保留字 **DESC**：

CREATE INDEX PersonIndex

ON Person (LastName DESC)

假如您希望索引不止一个列，您可以在括号中列出这些列的名称，用逗号隔开：

CREATE INDEX PersonIndex

ON Person (LastName, FirstName)

### **通过使用 DROP 语句，可以轻松地删除索引、表和数据库。**

DROP INDEX table\_name.index\_name

## SQL DROP TABLE 语句

DROP TABLE 语句用于删除表（表的结构、属性以及索引也会被删除）：

DROP TABLE 表名称

## SQL DROP DATABASE 语句

DROP DATABASE 语句用于删除数据库：

DROP DATABASE 数据库名称

## SQL TRUNCATE TABLE 语句

如果我们仅仅需要除去表内的数据，但并不删除表本身，那么我们该如何做呢？

请使用 TRUNCATE TABLE 命令（仅仅删除表格中的数据）：

TRUNCATE TABLE 表名称

## ALTER TABLE 语句

ALTER TABLE 语句用于在已有的表中添加、修改或删除列。

### SQL ALTER TABLE 语法

如需在表中添加列，请使用下列语法:

ALTER TABLE table\_name

ADD column\_name datatype

要删除表中的列，请使用下列语法：

ALTER TABLE table\_name

DROP COLUMN column\_name

**注释：**某些数据库系统不允许这种在数据库表中删除列的方式 (DROP COLUMN column\_name)。

要改变表中列的数据类型，请使用下列语法：

ALTER TABLE table\_name

ALTER COLUMN column\_name datatype

### ****Auto-increment 会在新记录插入表中时生成一个唯一的数字。****

## 用于 SQL Server 的语法

下列 SQL 语句把 "Persons" 表中的 "P\_Id" 列定义为 auto-increment 主键：

CREATE TABLE Persons

(

P\_Id int PRIMARY KEY IDENTITY,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

MS SQL 使用 IDENTITY 关键字来执行 auto-increment 任务。

默认地，IDENTITY 的开始值是 1，每条新记录递增 1。

要规定 "P\_Id" 列以 20 起始且递增 10，请把 identity 改为 IDENTITY(20,10)

要在 "Persons" 表中插入新记录，我们不必为 "P\_Id" 列规定值（会自动添加一个唯一的值）：

INSERT INTO Persons (FirstName,LastName)

VALUES ('Bill','Gates')

上面的 SQL 语句会在 "Persons" 表中插入一条新记录。"P\_Id" 会被赋予一个唯一的值。"FirstName" 会被设置为 "Bill"，"LastName" 列会被设置为 "Gates"。

### 什么是视图？

在 SQL 中，视图是基于 SQL 语句的结果集的可视化的表。

视图包含行和列，就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。我们可以向视图添加 SQL 函数、WHERE 以及 JOIN 语句，我们也可以提交数据，就像这些来自于某个单一的表。

**注释：**数据库的设计和结构不会受到视图中的函数、where 或 join 语句的影响。

### SQL CREATE VIEW 语法

CREATE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition

**注释：**视图总是显示最近的数据。每当用户查询视图时，数据库引擎通过使用 SQL 语句来重建数据。

## SQL CREATE VIEW 实例

可以从某个查询内部、某个存储过程内部，或者从另一个视图内部来使用视图。通过向视图添加函数、join 等等，我们可以向用户精确地提交我们希望提交的数据。

样本数据库 Northwind 拥有一些被默认安装的视图。视图 "Current Product List" 会从 Products 表列出所有正在使用的产品。这个视图使用下列 SQL 创建：

CREATE VIEW [Current Product List] AS

SELECT ProductID,ProductName

FROM Products

WHERE Discontinued=No

我们可以查询上面这个视图：

SELECT \* FROM [Current Product List]

Northwind 样本数据库的另一个视图会选取 Products 表中所有单位价格高于平均单位价格的产品：

CREATE VIEW [Products Above Average Price] AS

SELECT ProductName,UnitPrice

FROM Products

WHERE UnitPrice>(SELECT AVG(UnitPrice) FROM Products)

我们可以像这样查询上面这个视图：

SELECT \* FROM [Products Above Average Price]

另一个来自 Northwind 数据库的视图实例会计算在 1997 年每个种类的销售总数。请注意，这个视图会从另一个名为 "Product Sales for 1997" 的视图那里选取数据：

CREATE VIEW [Category Sales For 1997] AS

SELECT DISTINCT CategoryName,Sum(ProductSales) AS CategorySales

FROM [Product Sales for 1997]

GROUP BY CategoryName

我们可以像这样查询上面这个视图：

SELECT \* FROM [Category Sales For 1997]

我们也可以向查询添加条件。现在，我们仅仅需要查看 "Beverages" 类的全部销量：

SELECT \* FROM [Category Sales For 1997]

WHERE CategoryName='Beverages'

## SQL 更新视图

您可以使用下面的语法来更新视图：

SQL CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition

现在，我们希望向 "Current Product List" 视图添加 "Category" 列。我们将通过下列 SQL 更新视图：

CREATE VIEW [Current Product List] AS

SELECT ProductID,ProductName,Category

FROM Products

WHERE Discontinued=No

## SQL 撤销视图

您可以通过 DROP VIEW 命令来删除视图。

SQL DROP VIEW Syntax

DROP VIEW view\_name

## SQL 日期

当我们处理日期时，最难的任务恐怕是确保所插入的日期的格式，与数据库中日期列的格式相匹配。

只要数据包含的只是日期部分，运行查询就不会出问题。但是，如果涉及时间，情况就有点复杂了。

## SQL Server Date 函数

下面的表格列出了 SQL Server 中最重要的内建日期函数：

|  |  |
| --- | --- |
| **函数** | **描述** |
| [GETDATE()](https://www.w3school.com.cn/sql/func_getdate.asp) | 返回当前日期和时间 |
| [DATEPART()](https://www.w3school.com.cn/sql/func_datepart.asp) | 返回日期/时间的单独部分 |
| [DATEADD()](https://www.w3school.com.cn/sql/func_dateadd.asp) | 在日期中添加或减去指定的时间间隔 |
| [DATEDIFF()](https://www.w3school.com.cn/sql/func_datediff.asp) | 返回两个日期之间的时间 |
| [CONVERT()](https://www.w3school.com.cn/sql/func_convert.asp) | 用不同的格式显示日期/时间 |

在讨论日期查询的复杂性之前，我们先来看看最重要的内建日期处理函数。

SQL Server 使用下列数据类型在数据库中存储日期或日期/时间值：

* DATE - 格式 YYYY-MM-DD
* DATETIME - 格式: YYYY-MM-DD HH:MM:SS
* SMALLDATETIME - 格式: YYYY-MM-DD HH:MM:SS
* TIMESTAMP - 格式: 唯一的数字

## 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。

**注释：**无法比较 NULL 和 0；它们是不等价的。

## SQL IS NULL

我们如何仅仅选取在 "Address" 列中带有 NULL 值的记录呢？

我们必须使用 IS NULL 操作符：

SELECT LastName,FirstName,Address FROM Persons

WHERE Address IS NULL

## SQL IS NOT NULL

我们如何选取在 "Address" 列中不带有 NULL 值的记录呢？

我们必须使用 IS NOT NULL 操作符：

SELECT LastName,FirstName,Address FROM Persons

WHERE Address IS NOT NULL

### **现代的 SQL 服务器构建在 RDBMS 之上。**

## RDBMS - 关系数据库管理系统（Relational Database Management System）

关系数据库管理系统 (RDBMS) 也是一种数据库管理系统，其数据库是根据数据间的关系来组织和访问数据的。

20 世纪 70 年代初，IBM 公司发明了 RDBMS。

RDBMS 是 SQL 的基础，也是所有现代数据库系统诸如 Oracle、SQL Server、IBM DB2、Sybase、MySQL 以及 Microsoft Access 的基础。

## 函数的语法

内建 SQL 函数的语法是：

SELECT function(列) FROM 表

## 定义和用法

AVG 函数返回数值列的平均值。NULL 值不包括在计算中。

### SQL AVG() 语法

SELECT AVG(column\_name) FROM table\_name

**COUNT() 函数返回匹配指定条件的行数。**

## SQL COUNT() 语法

### SQL COUNT(column\_name) 语法

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

SELECT COUNT(column\_name) FROM table\_name

### SQL COUNT(\*) 语法

COUNT(\*) 函数返回表中的记录数：

SELECT COUNT(\*) FROM table\_name

### SQL COUNT(DISTINCT column\_name) 语法

COUNT(DISTINCT column\_name) 函数返回指定列的不同值的数目：

SELECT COUNT(DISTINCT column\_name) FROM table\_name

**注释：**COUNT(DISTINCT) 适用于 ORACLE 和 Microsoft SQL Server，但是无法用于 Microsoft Access。

## FIRST() 函数

FIRST() 函数返回指定的字段中第一个记录的值。

**提示：**可使用 ORDER BY 语句对记录进行排序。

### SQL FIRST() 语法

SELECT FIRST(column\_name) FROM table\_name

## LAST() 函数

LAST() 函数返回指定的字段中最后一个记录的值。

**提示：**可使用 ORDER BY 语句对记录进行排序。

### SQL LAST() 语法

SELECT LAST(column\_name) FROM table\_name

## MAX() 函数

MAX 函数返回一列中的最大值。NULL 值不包括在计算中。

### SQL MAX() 语法

SELECT MAX(column\_name) FROM table\_name

**注释：**MIN 和 MAX 也可用于文本列，以获得按字母顺序排列的最高或最低值。

### MIN函数也是一样

## SUM() 函数

SUM 函数返回数值列的总数（总额）。

### SQL SUM() 语法

SELECT SUM(column\_name) FROM table\_name

**合计函数 (比如 SUM) 常常需要添加 GROUP BY 语句。**

## GROUP BY 语句

GROUP BY 语句用于结合合计函数，根据一个或多个列对结果集进行分组。

### SQL GROUP BY 语法

SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

WHERE column\_name operator value

GROUP BY column\_name

## SQL GROUP BY 实例

我们拥有下面这个 "Orders" 表：

|  |  |  |  |
| --- | --- | --- | --- |
| **O\_Id** | **OrderDate** | **OrderPrice** | **Customer** |
| 1 | 2008/12/29 | 1000 | Bush |
| 2 | 2008/11/23 | 1600 | Carter |
| 3 | 2008/10/05 | 700 | Bush |
| 4 | 2008/09/28 | 300 | Bush |
| 5 | 2008/08/06 | 2000 | Adams |
| 6 | 2008/07/21 | 100 | Carter |

现在，我们希望查找每个客户的总金额（总订单）。

我们想要使用 GROUP BY 语句对客户进行组合。

我们使用下列 SQL 语句：

SELECT Customer,SUM(OrderPrice) FROM Orders

GROUP BY Customer

结果集类似这样：

|  |  |
| --- | --- |
| **Customer** | **SUM(OrderPrice)** |
| Bush | 2000 |
| Carter | 1700 |
| Adams | 2000 |

很棒吧，对不对？

让我们看一下如果省略 GROUP BY 会出现什么情况：

SELECT Customer,SUM(OrderPrice) FROM Orders

结果集类似这样：

|  |  |
| --- | --- |
| **Customer** | **SUM(OrderPrice)** |
| Bush | 5700 |
| Carter | 5700 |
| Bush | 5700 |
| Bush | 5700 |
| Adams | 5700 |
| Carter | 5700 |

上面的结果集不是我们需要的。

那么为什么不能使用上面这条 SELECT 语句呢？解释如下：上面的 SELECT 语句指定了两列（Customer 和 SUM(OrderPrice)）。"SUM(OrderPrice)" 返回一个单独的值（"OrderPrice" 列的总计），而 "Customer" 返回 6 个值（每个值对应 "Orders" 表中的每一行）。因此，我们得不到正确的结果。不过，您已经看到了，GROUP BY 语句解决了这个问题。

## GROUP BY 一个以上的列

我们也可以对一个以上的列应用 GROUP BY 语句，就像这样：

SELECT Customer,OrderDate,SUM(OrderPrice) FROM Orders

GROUP BY Customer,OrderDate

## HAVING 子句

在 SQL 中增加 HAVING 子句原因是，WHERE 关键字无法与合计函数一起使用。

### SQL HAVING 语法

SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

WHERE column\_name operator value

GROUP BY column\_name

HAVING aggregate\_function(column\_name) operator value

我们在 SQL 语句中增加了一个普通的 WHERE 子句：

SELECT Customer,SUM(OrderPrice) FROM Orders

WHERE Customer='Bush' OR Customer='Adams'

GROUP BY Customer

HAVING SUM(OrderPrice)>1500

## UCASE() 函数

UCASE 函数把字段的值转换为大写。

### SQL UCASE() 语法

SELECT UCASE(column\_name) FROM table\_name

## LCASE() 函数

LCASE 函数把字段的值转换为小写。

### SQL LCASE() 语法

SELECT LCASE(column\_name) FROM table\_name

## MID() 函数

MID 函数用于从文本字段中提取字符。

### SQL MID() 语法

SELECT MID(column\_name,start[,length]) FROM table\_name

### 参数描述column\_name必需。要提取字符的字段。start必需。规定开始位置（起始值是 1）。length可选。要返回的字符数。如果省略，则 MID() 函数返回剩余文本。

图形用户界面, 文本, 应用程序

描述已自动生成我们使用如下 SQL 语句：

SELECT MID(City,1,3) as SmallCity FROM Persons

## LEN() 函数

LEN 函数返回文本字段中值的长度。

### SQL LEN() 语法

SELECT LEN(column\_name) FROM table\_name

## ROUND() 函数

ROUND 函数用于把数值字段舍入为指定的小数位数。

### SQL ROUND() 语法

SELECT ROUND(column\_name,decimals) FROM table\_name

## NOW() 函数

NOW 函数返回当前的日期和时间。

**提示：**如果您在使用 Sql Server 数据库，请使用 getdate() 函数来获得当前的日期时间。

### SQL NOW/GETDATE() 语法

SELECT GETdate() FROM table\_name

## FORMAT() 函数

FORMAT 函数用于对字段的显示进行格式化。

### SQL FORMAT() 语法

SELECT FORMAT(column\_name,format) FROM table\_name

### 

|  |  |
| --- | --- |
| **参数** | **描述** |
| column\_name | 必需。要格式化的字段。 |
| format | 必需。规定格式。 |

SELECT ProductName, UnitPrice, FORMAT(Now(),'YYYY-MM-DD') as PerDate

FROM Products

### 电脑萤幕的截图 描述已自动生成