数据库名：wangct 表名：wtable

创建数据库： create database wangct

删除数据库： drop database wangct

创建表： create table wtable(id int not null primary key auto\_increment,name varchar(255) not null)

删除表： drop table wtable

插入数据： insert into wtable(name,age) values(‘wangchuitong’,23)

查询数据： select name,age from wtable

添加条件查询： select \* from wtable where id = 1

更新数据： update wtable set name = ‘wangchuitong’,age = 2 where id = 1

删除数据： delete from wtable where id = 1

模糊查询: select \* from wtable where name like ‘%wang%’

排序： select \* from wtbale order by name asc（升序）desc（降序）

多表查询： select a.name,b.age from wtable a,wtable2 b where a.id = 1 and b.id = 1

NULL值判断： is null ， is not null

正则表达式： select \* from wtable where name regexp ‘^wangct’

删除表字段： alter table wtable drop name

修改表字段： alter table wtable modify name varchar(500)

修改表名： alter table wtable rename to wtable2

创建索引： alter table wtable add index indexName(name)

删除索引： drop index[indexName] on wtable

显示指定表创建语句： show create table wtable

复制数据： insert into wtable2(name,age) select name,age from wtable

过滤重复数据： select distinct \* from wtable

时间函数：

函数 描述

NOW() 返回当前的日期和时间

CURDATE() 返回当前的日期

CURTIME() 返回当前的时间

DATE() 提取日期或日期/时间表达式的日期部分

EXTRACT() 返回日期/时间按的单独部分

DATE\_ADD() 给日期添加指定的时间间隔

DATE\_SUB() 从日期减去指定的时间间隔

DATEDIFF() 返回两个日期之间的天数

DATE\_FORMAT() 用不同的格式显示日期/时间

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() 返回当前日期和时间

DATEPART() 返回日期/时间的单独部分

DATEADD() 在日期中添加或减去指定的时间间隔

DATEDIFF() 返回两个日期之间的时间

CONVERT() 用不同的格式显示日期/时间

查询

select \* from user where id = 1; //从表user中查询所有id=1的数据

select id from user where id =2; //从表user中查询所有id=1的数据，数据只有一个id字段

//获取两个表的数据，过滤条件为id在另一个表存在

select \* from user u join dynamic d on u.id = d.id;

select:

SELECT 列名称 FROM 表名称

SELECT LastName,FirstName FROM Persons

distinct:

//过滤相同的值

SELECT DISTINCT 列名称 FROM 表名称

where:

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 在某个范围内

LIKE 搜索某种模式

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。

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

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

insert into:

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,....)

update:

Update 语句用于修改表中的数据。

语法：

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

delete:

DELETE 语句用于删除表中的行。

语法

DELETE FROM 表名称 WHERE 列名称 = 值

TRUNCATE TABLE table1 //删除数据，并重置自增

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

MySQL 语法

SELECT column\_name(s)

FROM table\_name

LIMIT number

例子

SELECT \*

FROM Persons

LIMIT 5

Oracle 语法

SELECT column\_name(s)

FROM table\_name

WHERE ROWNUM <= number

例子

SELECT \*

FROM Persons

WHERE ROWNUM <= 5

like:

//%表示通配符

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%'

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%'

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

[charlist] 字符列中的任何单一字符

[^charlist]

或者 不在字符列中的任何单一字符

[!charlist]

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,...)

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')

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

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'

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'

除了我们在上面的例子中使用的 INNER JOIN（内连接），我们还可以使用其他几种连接。

下面列出了您可以使用的 JOIN 类型，以及它们之间的差异。

JOIN: 如果表中有至少一个匹配，则返回行

LEFT JOIN: 即使右表中没有匹配，也从左表返回所有的行

RIGHT JOIN: 即使左表中没有匹配，也从右表返回所有的行

FULL JOIN: 只要其中一个表中存在匹配，就返回行

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

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

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 SELECT INTO 语句可用于创建表的备份复件。

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

我们必须使用 IS NULL 操作符：

SELECT LastName,FirstName,Address FROM Persons

WHERE Address IS NULL

结果集：

LastName FirstName Address

Adams John

Carter Thomas

提示：请始终使用 IS NULL 来查找 NULL 值。

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

在 MySQL 中，有三种主要的类型：文本、数字和日期/时间类型。

Text 类型：

数据类型 描述

CHAR(size) 保存固定长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的长度。最多 255 个字符。

VARCHAR(size)

保存可变长度的字符串（可包含字母、数字以及特殊字符）。在括号中指定字符串的最大长度。最多 255 个字符。

注释：如果值的长度大于 255，则被转换为 TEXT 类型。

TINYTEXT 存放最大长度为 255 个字符的字符串。

TEXT 存放最大长度为 65,535 个字符的字符串。

BLOB 用于 BLOBs (Binary Large OBjects)。存放最多 65,535 字节的数据。

MEDIUMTEXT 存放最大长度为 16,777,215 个字符的字符串。

MEDIUMBLOB 用于 BLOBs (Binary Large OBjects)。存放最多 16,777,215 字节的数据。

LONGTEXT 存放最大长度为 4,294,967,295 个字符的字符串。

LONGBLOB 用于 BLOBs (Binary Large OBjects)。存放最多 4,294,967,295 字节的数据。

ENUM(x,y,z,etc.)

允许你输入可能值的列表。可以在 ENUM 列表中列出最大 65535 个值。如果列表中不存在插入的值，则插入空值。

注释：这些值是按照你输入的顺序存储的。

可以按照此格式输入可能的值：ENUM('X','Y','Z')

SET 与 ENUM 类似，SET 最多只能包含 64 个列表项，不过 SET 可存储一个以上的值。

Number 类型：

数据类型 描述

TINYINT(size) -128 到 127 常规。0 到 255 无符号\*。在括号中规定最大位数。

SMALLINT(size) -32768 到 32767 常规。0 到 65535 无符号\*。在括号中规定最大位数。

MEDIUMINT(size) -8388608 到 8388607 普通。0 to 16777215 无符号\*。在括号中规定最大位数。

INT(size) -2147483648 到 2147483647 常规。0 到 4294967295 无符号\*。在括号中规定最大位数。

BIGINT(size) -9223372036854775808 到 9223372036854775807 常规。0 到 18446744073709551615 无符号\*。在括号中规定最大位数。

FLOAT(size,d) 带有浮动小数点的小数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。

DOUBLE(size,d) 带有浮动小数点的大数字。在括号中规定最大位数。在 d 参数中规定小数点右侧的最大位数。

DECIMAL(size,d) 作为字符串存储的 DOUBLE 类型，允许固定的小数点。

\* 这些整数类型拥有额外的选项 UNSIGNED。通常，整数可以是负数或正数。如果添加 UNSIGNED 属性，那么范围将从 0 开始，而不是某个负数。

Date 类型：

数据类型 描述

DATE()

日期。格式：YYYY-MM-DD

注释：支持的范围是从 '1000-01-01' 到 '9999-12-31'

DATETIME()

\*日期和时间的组合。格式：YYYY-MM-DD HH:MM:SS

注释：支持的范围是从 '1000-01-01 00:00:00' 到 '9999-12-31 23:59:59'

TIMESTAMP()

\*时间戳。TIMESTAMP 值使用 Unix 纪元('1970-01-01 00:00:00' UTC) 至今的描述来存储。格式：YYYY-MM-DD HH:MM:SS

注释：支持的范围是从 '1970-01-01 00:00:01' UTC 到 '2038-01-09 03:14:07' UTC

TIME() 时间。格式：HH:MM:SS 注释：支持的范围是从 '-838:59:59' 到 '838:59:59'

YEAR()

2 位或 4 位格式的年。

注释：4 位格式所允许的值：1901 到 2155。2 位格式所允许的值：70 到 69，表示从 1970 到 2069。

\* 即便 DATETIME 和 TIMESTAMP 返回相同的格式，它们的工作方式很不同。在 INSERT 或 UPDATE 查询中，TIMESTAMP 自动把自身设置为当前的日期和时间。TIMESTAMP 也接受不同的格式，比如 YYYYMMDDHHMMSS、YYMMDDHHMMSS、YYYYMMDD 或 YYMMDD。

SQL Server 数据类型

Character 字符串：

数据类型 描述 存储

char(n) 固定长度的字符串。最多 8,000 个字符。 n

varchar(n) 可变长度的字符串。最多 8,000 个字符。

varchar(max) 可变长度的字符串。最多 1,073,741,824 个字符。

text 可变长度的字符串。最多 2GB 字符数据。

Unicode 字符串：

数据类型 描述 存储

nchar(n) 固定长度的 Unicode 数据。最多 4,000 个字符。

nvarchar(n) 可变长度的 Unicode 数据。最多 4,000 个字符。

nvarchar(max) 可变长度的 Unicode 数据。最多 536,870,912 个字符。

ntext 可变长度的 Unicode 数据。最多 2GB 字符数据。

Binary 类型：

数据类型 描述 存储

bit 允许 0、1 或 NULL

binary(n) 固定长度的二进制数据。最多 8,000 字节。

varbinary(n) 可变长度的二进制数据。最多 8,000 字节。

varbinary(max) 可变长度的二进制数据。最多 2GB 字节。

image 可变长度的二进制数据。最多 2GB。

Number 类型：

数据类型 描述 存储

tinyint 允许从 0 到 255 的所有数字。 1 字节

smallint 允许从 -32,768 到 32,767 的所有数字。 2 字节

int 允许从 -2,147,483,648 到 2,147,483,647 的所有数字。 4 字节

bigint 允许介于 -9,223,372,036,854,775,808 和 9,223,372,036,854,775,807 之间的所有数字。 8 字节

decimal(p,s)

固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。

p 参数指示可以存储的最大位数（小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是 18。

s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。

5-17 字节

numeric(p,s)

固定精度和比例的数字。允许从 -10^38 +1 到 10^38 -1 之间的数字。

p 参数指示可以存储的最大位数（小数点左侧和右侧）。p 必须是 1 到 38 之间的值。默认是 18。

s 参数指示小数点右侧存储的最大位数。s 必须是 0 到 p 之间的值。默认是 0。

5-17 字节

smallmoney 介于 -214,748.3648 和 214,748.3647 之间的货币数据。 4 字节

money 介于 -922,337,203,685,477.5808 和 922,337,203,685,477.5807 之间的货币数据。 8 字节

float(n) 从 -1.79E + 308 到 1.79E + 308 的浮动精度数字数据。 参数 n 指示该字段保存 4 字节还是 8 字节。float(24) 保存 4 字节，而 float(53) 保存 8 字节。n 的默认值是 53。 4 或 8 字节

real 从 -3.40E + 38 到 3.40E + 38 的浮动精度数字数据。 4 字节

Date 类型：

数据类型 描述 存储

datetime 从 1753 年 1 月 1 日 到 9999 年 12 月 31 日，精度为 3.33 毫秒。 8 bytes

datetime2 从 1753 年 1 月 1 日 到 9999 年 12 月 31 日，精度为 100 纳秒。 6-8 bytes

smalldatetime 从 1900 年 1 月 1 日 到 2079 年 6 月 6 日，精度为 1 分钟。 4 bytes

date 仅存储日期。从 0001 年 1 月 1 日 到 9999 年 12 月 31 日。 3 bytes

time 仅存储时间。精度为 100 纳秒。 3-5 bytes

datetimeoffset 与 datetime2 相同，外加时区偏移。 8-10 bytes

timestamp 存储唯一的数字，每当创建或修改某行时，该数字会更新。timestamp 基于内部时钟，不对应真实时间。每个表只能有一个 timestamp 变量。

其他数据类型：

数据类型 描述

sql\_variant 存储最多 8,000 字节不同数据类型的数据，除了 text、ntext 以及 timestamp。

uniqueidentifier 存储全局标识符 (GUID)。

xml 存储 XML 格式化数据。最多 2GB。

cursor 存储对用于数据库操作的指针的引用。

table 存储结果集，供稍后处理。

函数

AVG 函数返回数值列的平均值。NULL 值不包括在计算中。

SELECT AVG(column\_name) FROM table\_name

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

现在，我们希望计算 "Orders" 表中不同客户的数目。

我们使用如下 SQL 语句：

SELECT COUNT(DISTINCT Customer) AS NumberOfCustomers FROM Orders

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

Min 函数返回一列中的最小值。NULL 值不包括在计算中。

SQL Min () 语法

SELECT Min (column\_name) FROM table\_name

SUM 函数返回数值列的总数（总额）。

SQL SUM() 语法

SELECT SUM(column\_name) FROM table\_name

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

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 函数把字段的值转换为小写。

SQL UCASE() 语法

SELECT lCASE (column\_name) FROM table\_name

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

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

参数 描述

column\_name 必需。要舍入的字段。

decimals 必需。规定要返回的小数位数。

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。

date\_format(date,’%Y-%m-%d %H:%i:%s’) //时间转化为字符串

%a 缩写星期名

%b 缩写月名

%c 月，数值

%D 带有英文前缀的月中的天

%d 月的天，数值(00-31)

%e 月的天，数值(0-31)

%f 微秒

%H 小时 (00-23)

%h 小时 (01-12)

%I 小时 (01-12)

%i 分钟，数值(00-59)

%j 年的天 (001-366)

%k 小时 (0-23)

%l 小时 (1-12)

%M 月名

%m 月，数值(00-12)

%p AM 或 PM

%r 时间，12-小时（hh:mm:ss AM 或 PM）

%S 秒(00-59)

%s 秒(00-59)

%T 时间, 24-小时 (hh:mm:ss)

%U 周 (00-53) 星期日是一周的第一天

%u 周 (00-53) 星期一是一周的第一天

%V 周 (01-53) 星期日是一周的第一天，与 %X 使用

%v 周 (01-53) 星期一是一周的第一天，与 %x 使用

%W 星期名

%w 周的天 （0=星期日, 6=星期六）

%X 年，其中的星期日是周的第一天，4 位，与 %V 使用

%x 年，其中的星期一是周的第一天，4 位，与 %v 使用

%Y 年，4 位

%y 年，2 位