SqlServer学习文档

**数据库的设计**

## 为什么需要设计数据库

良好的数据库设计 糟糕的数据库设计

节省数据的存储空间 数据冗余，存储空间浪费

能够保证数据的完整性 内存空间浪费

方便进行数据库应用系统的开发 数据更新和插入异常

## 软件项目开发周期中数据库设计

1.需求分析阶段：分析客户的业务和数据处理需求

①收集信息

与该系统有关人员进行交流，座谈，充分了解用户的需求，理解数据库需要完成的任务

②标识实体（Entitly）

标识数据库要管理的关键对象或实体、、、、，实体一般是名词

③标识每个实体的属性（Attribute）

④标识实体之间的关系（Relationship）

2.概要设计阶段：设计数据库的E-R模型图，确认需求信息的正确和完整

3.详细设计阶段：应用三大范式审核数据库结构

4.代码编写阶段：物理实现数据库，编码实现应用

5.软件测试阶段：……

E-R图

ER图也可以成为实体-关系图，它包括一些具有特定意义的图像符号，

以图形的方式将数据库的整个逻辑结构表示出来

符号 含义

长方形 □ 实体，一般是名词

椭圆形 ○ 属性，一般是名词

菱形 ◇ 关系，一般是动词

映射基数：实体和实体之间的关系

①一对一

②一对多

③多对一

④多对多

## 数据范式化

范式是具有最小冗余的表的结构，规范表里的结构

第一范式：(First Normal Formate)

第一范式的目标是确保每列的原子性

如果每列都是不可再分的最小数据单元（也成为最小的原子单元），则满足第一范式（1FN）

第二范式：(Second Normal Formate)

如果一个关系满足1NF，并且除了主键以外的其他列，都依赖于该主键，则满足第二范式(2FN)

第二范式要求每个表只描述一件事情。

第三范式：(Third Normal Formate)

如果一个关系满足2FN，并且除了主键以外的其他列都不传递依赖于主键列，

列于列之间不存在相互依赖的关系，则满足第三范式（3FN）

**数据库的实现**

## 数据库三大文件

主数据文件 .mdf

次数据文件 .ndf

日志文件 .ldf

## 数据库 DataBase

### 创建数据库的语法

//参数和参数之间必须用，隔开

Create DATABASE 数据库名

on primary //位于的工作组 primary是默认的，所以可以省略

(

//指定数据库的名称

name='StuDB\_date',

//指定数据库的位置

FileName='d:\project\MySchool\_date.mdf' ,

//指定数据库的初始值大小

size=5MB,

//指定数据库的最大值

Maxsize=100MB,

//指定数据库的增长值

Filegrowth=10%

),()--------------------------------//如果有次数据文件.ndf就在这直接打,()在括号内打创建内容。内容同上

log on

(

//指定数据库日志文件的名称

name='StuDB\_log',

//指定数据库的位置

FileName='d:\project\StuDb\_log.ldf',

//指定数据库的初始值大小

size=1MB,

//指定数据库的最大值

Maxsize=10MB,

//指定数据库的增长值

Filegrowth=10%

)

//完整案例--------------------------------------------判断MySchool数据库是否存在，如果存在就删除，然后创建

use master

if exists(select \* from sysdatabases where name='MySchool')

drop database MySchool

create database MySchool

on primary

(

name='MySchool\_data.mdf',

filename='d:\project\MySchool\_data.mdf',

size=5MB,

maxsize=100MB,

filegrowth=15%

)

log on

(

name='MySchool\_log.ldf',

filename='d:\project\MySchool\_log.ldf',

size=2MB,

filegrowth=1MB

)

master存储所有数据库的信息，建库建表必须在master下面运行，如果怕忘可以在最前面加上use master

### 创建表的语法

use MySchool --将当前数据库设置为myschool数据库

--判断表是否存在，存在就删除

if exists(select \* from sysobjects where name='Student')

drop table Student --删除表

Go

create table Student

(

StudentNo int not null, --学号，非空

loginPwd nvarchar(20) not null, --密码，非空

StudentName nvarchar(20) not null, --姓名，非空

Sex bit not null, --性别，非空

GradeId int not null, --年级号，非空

Phone nvarchar(50) not null, --电话，非空

BornDate datetime not null, --出生日期，非空

Address nvarchar(255), --地址

Email nvarchar(50), --邮件证号

IDENTITYCard varchar(18) --身份证号

)

GO

标识列:identity(1,1), 放到not null后面，设置这一列为标识列，()内前面是标识种子，后面是增长

--------------Go是把数据分成一段一段的!

### 创建列的语法

--------在已有的表中添加列

alter table 表名

add 列名 数据类型 可以加主键表示列等约束

--------例

alter table student

add BTotal int null

--------在已有的表中删除咧

alter table 表名

drop colunm 列名

--------例

alter table student

drop column BTotal

### 实施约束的目的

实施约束的目的：确保表中数据的完整性。

实体完整性约束：

能够唯一标识表中的每一条记录,

约束行的取值

实现方式：主键，唯一键，IDENTITY属性

域完整性约束：

约束某列的取值范围

引用完整性约束：

用一个表来限制另一个表中某列的取值范围

自定义完整性约束：

约束某个单元格的内容

常用的约束类型有： Constraint 约束的意思

主键约束：(Primary Key Constraint)

要求主键列数据唯一

唯一约束：(Unique Constraint)

要求该列数据唯一，允许为空，但只能出现一个空值

检查约束：(Check Consyraint)

某列取值范围限制，格式限制等

默认约束：(default Constraint)

某列的默认值

外键约束：(Foreign Key Constraint)

用于量表之间建立关系，需要指定引用主表的哪列

主键约束与唯一约束的区别

1.主键约束所在的列不允许有空值，唯一约束所在的列允许有空值

2.每个表中可以有一个主键，多个唯一键

### 添加约束的语法

ALTER TABLE 表明

Add constraint 约束名 约束类型 具体的约束说明

约束名的取名规则：约束类型\_约束列

主键(Primary Key)约束：如PK\_StuNo 复合主键。列名和列名之间用,隔开

唯一(Unique)约束：如UK\_StuNo

默认(default)约束：如DK\_StuNo

检查(Check)约束：如CK\_StuNo

外键(Foreign)约束：如FK\_StuNo

非空约束：没有名字。用检查约束写：如CK\_StuNo

删除约束的语法：

Alter table 表名

Drop Constraint 约束名

-With noCheck 不检查数据。

在Alter table student With noCheck

就是你如果表里有数据，你后期想补添约束，但是之前有数据不符合，就用这种方法，约束后来写入的数据,不约束以前写的数据！！

注意：创建表结构后应立刻添加约束，不要马上插入数据，以避免插入的数据不符合约束要求，保证表中数据满足约束要

例：

use MySchool

--添加主键约束

alter table student

add Constraint PK\_StudentNo primary Key(StudentNo)

--唯一约束：身份证号

alter table student

add Constraint UK\_IdentityCard Unique (IdentityCard)

--默认约束：Address列默认值是“地址不详”

alter table student

add Constraint DF\_Address default ('地址不详') for address

--检查约束:出生日期的值为1980年1月1日后

alter table student

add Constraint CK\_BornDate check (borndate>='1980-1-1')

--外键约束：

alter table student

add Constraint FK\_Grade foreign key (GradeID) references Grade(GradeID)

--删除约束

alter table student

drop constraint DF\_Address

--非空约束

alter table Subject

add Constraint Ck\_SubjectName check (SubjectName is not null)

--复合主键

alter table Result

add Constraint PK\_FuHe primary key(StudentNo,SubjectNO,ExamDate)

**SQL编程**

## 局部变量

局部变量必须以标记@作为前缀，如@age

局部变量的使用也是先声明,再赋值

声明局部变量

DECLARE @变量名 数据类型

注意：

1.变量名严格区分大小写！如同java和C#

2.在T-SQL不允许出现分号，声明变量时无标点符号

例子:

declare @Name varchar(8)

赋值

set @变量名=值

或者

select @变量名=值

例子:

set @name='张三'

select @name=studentName From student where student ='10011'

### set和select的区别

1.使用场合不一样，set是把固定的值赋值给变量，select是把从表中查询出来的值赋值给变量

2.set查询多个值的时候出错

select查询多个值的时候把最后一个值赋值给变量

set和select的区别

set select

赋值的类型 一般用于给变量置顶的数据常量 一般用于从表中查询数据，然后赋值

同时对多个变量赋值 不支持 支持

表达式返回多个值时 出错 将返回的最后一个赋值给变量

表达式未返回值时 变量被赋NULL值 变量保持原值

declare @Age int //声明一个变量age

set @Age=21 //赋值

print @Age //输出

## 全局变量

变量 含义

@@ERROR 最后一个T-SQL错误的错误号 ★

@@IDENTITY 最后一次插入的标识值 ★

@@LANGUAGE 当前使用的语言的名称

@@MAX\_CONNECTIONS 可以创建的同时连接的最大数目

@@ROWCOUNT 受上一个SQL语句影响的行数

@@SERVERNAME 本地服务器的名称 ★

@@TRANSCOUNT 当前连接打开的事务数

@@VERSION SQL Server的版本信息 ★

print 是打印输出到控制台

语法：

Print 局部变量 或字符串

例：

Print +’服务器的名称：’+@@Servername

declare @Age int

set @Age=21

//把后面的值转换成前面的类型，用Convert()

convert(数据类型，表达式，样式) --样式是日期函数中，第三个参数苦于省略

--不同的样式使转换后的字符数据的显示格式不同

例：

convert(varchar(50),borndate,102)

convert(数据类型，表达式，样式)

--把int类型的Age 21转换为varchar类型的'21'

convert(varchar(5),@Age)

CAST用AS转换把前面的转换为后面的类型

cast(表达式 as 数据类型)

cast(getdate() as varchar(20))

convert()和cast()的不同点是，可以指定转换的样式

## 分支结构

begin end等同java和C#中的{}

if()

begin

end

else

begin

end

----------------------例

if(@result>85)

begin

print @Name+' 成绩优秀'

end

else if(@result>70)

begin

print @Name+' 成绩良好'

end

else if(@result>60)

begin

print @Name+ ' 成绩中等'

end

else

begin

print @Name+' 成绩良差'

end

## 循环结构

while循环结构

while()

begin

end

同样有break和continue

## CASE -END 多分支语句

--case

-- when 条件1 then 结果1

-- when 条件2 then 结果2

-- else

-- --其他结果

-- end

--------------------------------例

use MySchool

declare @subNO int, --科目编号

@Time datetime -----最近一次考试的时间

SELEct @subNO=SubjectNO from Subject where SubjectName='Java logic'

select @Time=MAX(ExamDate) from Result where SubjectNo=@subNO

select StudentNo as 学生编号,

case

when StudentResult between 90 and 100 then 'A'

when StudentResult between 80 and 89 then 'B'

when StudentResult between 70 and 79 then 'C'

when StudentResult between 60 and 69 then 'D'

else 'E'

end as 等级

from Result

where ExamDate=@Time and SubjectNo=@subNO

------------------------GO

GO要单独一行，只能和T-SQL语句一起使用

## 子查询

--1.子查询的结果当作SQL语句的条件

--注意：子查询的sql语句必须被放到小括号里面，并且执行时优先执行

--子查询和付查询的条件用比较运算符一起匹配，使用的时候子查询的结果只能有一个值。多个报错

-- 运算符（>,<,=,<=,>=）

--子查询可以插入到任何查询语句当中

--例：

---查询比李斯文小的学生的所有信息

select \* from Student

where BornDate>(

select BornDate

from Student

where StudentName='李斯文'

)

--一般来说，子连接查询可以替换表连接查询，但子查询不能被表连接替换

--子查询比较灵活，方便，常作为增删改查筛选条件，而表连接更适合于查看多表的数据

### IN关键字

--当子查询查询出的结果有多个值，可以用IN关键字(=换成in)

--意思是在多个值中都可以

### NOT IN关键字

--子查询查询出多个值，用NOT IN就代表不在查询出来的内容中所有的东西

### if exists子查询

if exists(子查询) --如果子查询的内容非空则返回true,否则返回false

begin

end

else

begin

end

### if NOT exists子查询

if not exists(子查询) --如果子查询的内容非空则返回true,否则返回false

begin

end

else

begin

end

### 分页查询的公式

--select top 每页显示的条数 \*

--from 表名

--where 主键列 not in

-- (

-- select top 0\*页码+(页码-1)\*条数 主键列 from 表名

-- )、

--order by 主键

**事务，视图和索引**

## 事务

为什么使用事务：事务能保持数据的完整性

事务是什么：

事务(transaction)是作为单个逻辑工作单元执行的一系列操作

多个操作作为一个整体向系统提交，要么都执行，要么都不执行

事务是一个不可分割的工作逻辑单元

事务的特点:

事务必须具备以下四个属性，简称ACID属性：

原子性(Atomicity)

事务是一个完整的操作，事务的各步操作是不可分开的，要么都执行，要么都不执行

一致性(Consistency)

当事务完成时，数据必须处于一致的状态

隔离性(lsolation)

并发事务之间彼此隔离，独立，它不应以任何方式依赖或影响其他事务

永久性(Durability)

事务完成后，他对数据库的修改永久保持

如何使用事务：使用SQL语句管理事务

开始事务

begin transaction

提交事务

commit transaction

回滚(撤销)事务

rollback transaction

事务中每一条sql执行完毕之后都要统计以下错误，最后如果错误大于0，必定有sql语句出错

### 事务分类

用begin tarnsaction 明确指定事务的开始

隐式事务

通过设置set implicit\_transaction on 语句，将隐式事务模式设置为打开，当以隐式事务操作时候，sql server将在提交或者回滚事务后自动启动更新事务

自动提交事务

这是sql server默认模式，它将每条sql语句视为一个事务，如果成功执行，则自动提交，如果错误则自动滚回，

实际开发时最常用的就是显示事务，它明确的指定事务开始的边界

## 视图

视图是一张虚拟表

表示一张表的部分数据或多张表的综合数据

其结构和数据是建立在对表的查询基础上

视图中不存放数据

数据存放在视图所引用的原始表中

一个原始表，根据不同用户的不同需求，可以创建不同的视图

--例如把源表删了，视图中数据就没了

if exists(SELECT \* FROM sys.objects where name ='视图名')

语法

create view 视图名

as

select语句

删除语法

drop view 视图名

视图可以使用多个表=

一个视图可以嵌套另一个视图

视图定义中的select语句不能包括以下内容：

1.不能使用order by 字句，除非在select语句中选择列表中也有一个top字句

2.不能使用into关键字

3.不能引用临时表或表变量

## 索引

索引的作用：通过使用索引，可以大大提高数据库的检索速度，改善数据库性能，优化数据库

索引：是SQL sever 编排数据的内部方法，他为sql server提供一种方法来编排查询数据

索引页：在数据库中存储索引的数据页，类似于汉语字典中的目录页

索引语法

create unique

index 索引名

on 表名(列名)

create unique

index ix\_student

on student (StudentNo)

删除

drop index 表名.索引名

drop index student.ix\_student

删除表的时候，里面所有索引同时被删除

if exists(select name from sysindexes where name='索引名称')

查询时使用索引

select 语句

with(index=索引名)

条件

select \* from Student

WITH (INDEX=ix\_student)

where StudentName like '张%'

### 操作索引

查看索引

EXEC sys.sp\_helpindex 表名/视图名

--查看result表的索引信息

EXEC sys.sp\_helpindex STUDENT

--查看数据库中全部索引的信息

USE mySchool

SELECT \* FROM sys.indexes

--创建和使用索引

--B数索引：默认的索引类型

create index index\_tbl\_emp\_empname

on tbl\_emp(empname);

--创建索引的语法：

create [索引类型] index 索引名称

on 表名(列名);

--删除索引的语法：

drop index 索引名称

1.导出数据

语法：exp 用户名/密码@数据库实例名称 file=导出文件路径.dmp full=y

示例：exp bdqn\_myschool/bdqn123@ORCL file=G:\bdqn\_myschool.dmp full=y

2.导入数据

语法：imp 用户名/密码@数据库实例名称 file=导入文件路径.dmp ignore=y

示例：imp bdqn\_myschool/bdqn123@ORCL file=G:\bdqn\_myschool.dmp ignore=y

备注：Oracle cmd命令可导出序列

如果使用PLSQL导出数据，则不会导出序列，所以导入的数据也不会有序列

如果使用cmd命令导出数据，则可以导出序列，所以导入的数据也会有序列

推荐：公司开发备份数据时，推荐使用cmd命令导出数据

导入数据时，可使用cmd命令导入，也可使用PLSQL导入

### 索引的优点缺点

优点

1.加快访问速度

2.加强行的唯一性

缺点

1.带索引的表在数据库中需要更多的存储空间

2.操作数据的命令需要更长的处理时间，因为他们需要对索引进行更新

按照下列标准选择建立索引的列

1.频繁搜索的列

2.经常用作查询的列

3.经常排序，分组的列

4.经常用作连接的列(主外键)

5.仅包含几个不同值的列

6.表中仅包含几行，为小型表创建索引可能不太划算，因为sql server中在索引中查询数据比在表中逐行搜索花的时间更长

经验:

1.查询时减少\*返回全部列，不要返回不需要的列

2.索引应尽量小，在字节数小的列上建立索引

3.where 字句中有多个表达式，包含索引列的表达式应置于其他条件表达式之前

4.避免order by字句中使用表达式

5.根据业务数据发送频率， 定期重新生成或重新组织索引，进行碎片整理

### 索引分类

--1.B数索引：oracle中默认索引类型，以树形结构的方式存储数据，数据都集中在树形结构末端的叶子上

--2.位图索引：低基数，适用于列的总数目大于列值的数目

--3.反向键索引：B数索引的一种特殊方式，反向每个列值如：1234 4321

--4.唯一索引、非唯一索引：每列的值是否相同

--5.单列索引、组合索引：在单个列上或多个列上创建索引

--6.基于函数的索引：在这一列上使用函数所创建的索引

**存储过程**

**什么是存储过程**

1预先存储好的SQL程序

2保存在SQL Server中

3通过名称和参数执行

**存储过程的优点**

1执行速度快

2允许模块化的程序射击

3提高系统安全性

4减少网络流通量

**存储过程的分类**

## 系统存储过程

系统存储过程

系统存储过程名称一般以'sp\_'开头

由SQ;Server创建、管理和使用

存放在Resource数据库中

类似C#语言类库中的方法

EXEC：执行

procedure ：存储过程

sp\_databases 列出服务器上所有数据库

EXEC sp\_databases

sp\_helpdb 报告有关制定数据库或所有数据库的信息

EXEC sp\_helpdb 或者

EXEC sp\_helpdb MySchool

sp\_renamedb 更改数据库的名称

EXEC sp\_renamedb 'MySchool','youSchool'

sp\_tables 返回当前环境下可查询的对象列表

EXEC sp\_tables

sp\_columns 返回某个表的列的信息

USE myschool

EXEC sys.sp\_columns Result

sp\_help 返回某个表的所有信息

EXEC sp\_help student

sp\_helpconstraint 返回某个表的所有约束

EXEC sys.sp\_helpconstraint student

sp\_helpindex 返回某个表的所有索引

EXEC sys.sp\_helpindex student

sp\_stored\_procedures 列出当前环境中所有存储过程

EXEC sys.sp\_stored\_procedures

sp\_password 添加或修改登录账户的密码

EXEC sys.sp\_password 'sa','sa'

sp\_helptext 显示默认值、未加密的存储过程、用户

定义的存储过程，触发器或视图的实际文本

## 用户自定义存储过程

--user system procedure ---usp

1.判断存储过程是否存在

USE myschool ---必须有

IF exists(SELECT \* FROM sysobjects where name ='usp\_test')

2.如果存在就删除

drop PROC usp\_test

3.创建存储过程---必须放到2个GO中，批处理里面

GO

CREATE PROC usp\_test

as

select \* from Student

GO

4.使用存储过程--必须放到批处理外面

EXEC usp\_test

## 扩展存储过程

扩展存储过程名称一般以'xp\_'开头

使用编辑语言(如C#)创建的外部存储过程

以DLL形式单独存

## 存储过程的参数

### 输入参数

向存储过程传入值

CREATE procedure usp\_unpass

@subName varchar(50), --科目名称 这两个为参数。

@score int=70 --及格分数--直接=70是默认值，吊用时候可用default

as

select \* from …………

调用语法 exec usp\_unpass 'C# OOP',70

或者 exec usp\_unpass @subName='C# OOP',@score=70

或者 exec usp\_unpass @subName='C# OOP'

### 输出参数

CREATE proc usp\_kaoShi

@zongRenShu int output, --考试总人数

--输出参数后面要加output

@weiTongGuo int output, --未及格的人数

@subName varchar(20),

@score int =99 --输入及格线默认为99分

AS

select…………………………

调用语法： --首先声明2个变量用来接受输出参数

DECLARE @a int,

@b int

EXEC usp\_kaoShi @a OUTPUT,@b OUTPUT,'java logic',77 --接受时也要加上output

PRINT '参加考试总人数：'+cast(@a as varchar(10))

print '未 通 过 人 数：'+cast(@b as varchar(10))

**SqlServer 语法结构**

## 创建数据表分为三步

1定义一个表的结构

2添加约束，保护数据完整性

3插入数据，在添加数据之前就要实施完整性约束

### 附加信息

什么是数据的完整性？

数据的真实性和有效性

创建数据表时如何保证数据的完整性？

实施完整性约束

## 实施完整性的方法有几种？分别是什么？

4种

1.域完整性约束 约束某列的取值范围

2.实体完整性约束 约束行的取值

3.自定义完整性约束 约束某个单元格的内容

4.引用完整性约束 用一个 表来限制另一个表的取值

## SQL语句分为几种？

1、插入单条数据sql

insert into 表名(列名，列名...) values(值,值...)

注：列名与值列表数量要一致

2、插入多条数据sql语句（3个）

//把一个表中数据放到另一个表中

1、insert into 表名(列名，列名...) select 列名、列名...from A表名

注：两个列名数量要对应。

//把表中的数据放到1个新表中

2、select 列名，列名 into 新表名 from 源表名

3、insert 表名(列名、列名) select'列值','列值' union select'列值','列值'

（注：列值要与列名数量对应）

3、查询sql语句+where条件

select \* from 表名 where 列名= 值

4、更新sql语句+where条件

update 表名 set 列名 = 更新值 where 列名 = 值

5、删除sql语句（两种方式）

(1)truncate table 表名

(注：不可以加where条件 )

(2)delete from 表名 where 列名 = 值

6、查询班级分数在60到80之间的学员有哪些

select \* from 表名 where 分数列 between 60 and 80

7、查询班级里面姓李的同学有哪些

select \* from 表名 where 姓名 like '%李%'

8、like和哪些通配符可以一起使用，并说出每个通配符的含义

(1)\_单个字符 (2)%任意长度字符

(3)[] 在中括号内满足 (4)[^] 不在中括号内满足

9、聚合函数有哪些，返回的 果是什么形式？

(1)avg()平均 (2)sum()求和 (3)count()次数 (4)max ()最大 (5)min()最小

注:聚合函数返回单行单列

10、模糊查询的三种查询关键字

1、 like

例： 只查询姓李的学生

select \* from 表名 where 列名 like '李%'

2、in

例： 查询学生地址在北京、上海、济南有那些人

select \* from 表名 where 列名 in ('北京','上海','济南')

3、between and

例：查询学生成绩在60到80分之间的

select \* from 表名 where 列名 between 60 and 80

## 多表连接查询

### 内连接(取并集数据)

1、select 列名 from 表名，表名　where 表名.列名=表名.列名

2、select 列名 from 表名 inner join 表名2 on 表名.列名=表名2.列名

### 外连接

1、左外连接(left左面为主表，要把主表中数据全部查询出来，主表存在数据，子表不存在关联数据，不存在关联的数据要以NULL形式填充)

select 列名 from 表名1 left join 表名2 on 表名1.列名=表名2.列名

2、右外连接（right右面为主表，要把主表中数据全部查询出来，主表存在数据，子表不存在关联数据，不存在关联的数据要以NULL形式填充）

select 列名 from 表名1 right join 表名2 on 表名1.列名=表名2.列名

## 函数

--首字母大写

select initcap('abcdefg') from dual

--Abcdefg

select initcap('ABCDEFG') from dual

--Abcdefg

--将字符串转换为小写

select lower('FFFFF') from dual

--将字符串转换为大写

select upper('FFFFF') from dual

--左裁剪

select LTRIM('AAAAFFFFF','AAAA') from dual

--右裁剪

select RTRIM('FFFFFAAAA','AAA') from dual

--字符串替换 把J替换为bl

select Replace('jack and jue','j','bl') from dual

---查找子串位置 取出第五位的字符

select instr('worldwide',5) from dual

--取子字符串 从第三个位置截取两位

select substr('abcdefg',3,2) from dual

--连接字符串

select concat('HELLO','WORLD') from dual

--取绝对值

select abs(-15) from dual

--向上取整

select ceil(14.5) from dual

--向下取整

select floor(14.5) from dual

--正弦

select sin(1.571) from dual

--余弦

select COS(9) from dual

--取符号

select sign(-15) from dual

--取余数

select mod(10,3) from dual

--m的n次幂

select power(3,3) from dual

--四舍五入

select round(100.756,2) from dual

--截断

select trunc(100.756,2) from dual

--平方根

select sqrt(4) from dual

--日期函数

--返回两个日期之间的月份

select months\_between('04-11月-05','11-1月-01') from dual

--将日期类型的字符串转换为指定格式的日期

to\_date(目标日期字符串，要转换的格式)

--将非日期类型的数据转换为日期类型

select to\_date('2017-03-30','yyyy-mm-dd') from dual

### 常用聚合函数

sum()求和,

max()最大值,

min()最小值,

count()计数,

avg()平均值