字符串函数

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 例子 |
| concat(S1,S2,··.Sn) | 字符串拼接，将S1,S2,···Sn拼接成一个字符串 | select concat('Hello' , ' MySQL'); |
| lower (str) | 将字符串str全部转为小写 | select lower('Hello'); |
| upper(str) | 将字符串str全部转为大写 | select upper('Hello'); |
| lpad(str,n,pad) | 左填充，用字符串pad对str的左边填充，达到n个字符串长度 | select lpad('01', 5, '-'); |
| rpad(str,n,pad) | 右填充，用字符串pad对str的右边填充，达到n个字符串长度 | select rpad('01', 5, '-'); |
| trim(str) | 去掉字符串头部和尾部的空格 | select trim(' Hello MySQL '); |
| substring(str,start,len) | 返回从字符串str从start位置起的1en个长度的字符串 | select substring('Hello MySQL',1,5); |

数值函数

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 例子 |
| cell(x) | 向上取整 | select ceil(1.1); |
| floor(x) | 向下取整 | select floor(1.9); |
| mod(x,y) | 返回x/y的模 | select mod(7,4); |
| rand() | 返回0~1的随机数 | select rand(); |
| round(x,y) | 求x四舍五入的值，保留y位小数 | select round(2.344,2); |

日期函数

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 例子 |
| curdate() | 返回当前日期 | select curdate(); |
| curtime() | 返回当前时间 | select curtime(); |
| now() | 返回当前日期和时间 | select now(); |
| year(date) | 获取指定date的年份 | select YEAR(now()); |
| month(date) | 获取指定date的月份 | select MONTH(now()); |
| day(date) | 获取指定date的日期 | select DAY(now()); |
| date\_add(date,interval exprtype) | 返回一个日期/时间值加上一个时间间隔expr后的时间值 | select date\_add(now(), INTERVAL 70 YEAR ); |
| datediff(date1,date2) | 返回起始时间date1和结束时间date2之间的天数 | select datediff('2021-10-01', '2021-12-01'); |

流程函数

|  |  |  |
| --- | --- | --- |
| 函数 | 功能 | 例子 |
| if(value,t,f) | 如果value为true，则返回t，否则返回f | select if(false, 'Ok', 'Error'); |
| ifnull(value1,value2) | 如果value1不为空，返回value1，否则返回value2 | select ifnull(null,'Default'); |
| case when [vall] then [res1] ...  else [default] end | 如果val1为true，返回res1，...  否则返回default默认值 |  |
| case [expr] when [vall] then [res1] ...  else [default] end | 如果expr的值等于val1，返回res1，...  否则返回default默认值 | select name,( case workaddress when '北京' then '一线城市' when '上海' then '一线城市'  else '二线城市' end ) as '工作地址' from emp; |

约束

|  |  |  |
| --- | --- | --- |
| 约束 | 描述 | 关键字 |
| 非空约束 | 限制该字段的数据不能为null | not null |
| 唯一约束 | 保证该字段的所有数据都是唯一、不重复的 | unique |
| 主键约束 | 主键是一行数据的唯一标识，要求非空且唯一 | primary key |
| 默认约束 | 保存数据时，如果未指定该字段的值，则采用默认值 | default |
| 检查约束 | 保证字段值满足某一个条件 | check |
| 外键约束 | 用来让两张表的数据之间建立连接，保证数据的一致性和完整性 | foreign key |

约束是作用于表中字段上的，可以在创建表/修改表的时候添加约束。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 | 约束条件 | 约束关键字 |
| id | ID唯一标识 | int | 主键自动增长 | prinary key,auto\_increment |
| name | 姓名 | varchar(10) | 不为空，且唯一 | not null,unigue |
| age | 年龄 | int | 大于0，小于等于120 | check |
| status | 状态 | char(1) | 无默认值，为1 | default |
| gender | 性别 | char(1) | 无 |  |

Create table tb\_user(

Id int auto\_increment prinary key comment ‘ID唯一标识’,

Name varchar(10) not null unigue comment ‘姓名’,

Age int cheak (age > 0 && age <= 120) comment ‘年龄’,

Status char(1) default ‘1’ comment ‘状态’,

Gender char(1) comment ‘性别’

);

外键约束添加与删除

Create table 表名(

字段名 数据类型,…… [constraint] [外键名称] foreign key (外键字段名) references 主表(主表列名)

);

Alter table 表名 add constraint 外键名称 foreign key (外键字段名) references主表 (主表列名) ;

Alter table 表名 drop foreign key 外键名称;

删除/更新行为

添加了外键之后，再删除父表数据时产生的约束行为

|  |  |
| --- | --- |
| 行为 | 说明 |
| no action | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。(与restrict一致) 默认行为 |
| restrict | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除/更新。(与no actiom一致) 默认行为 |
| cascade | 当在父表中删除/更新对应记录时，首先检查该记录是否有对应外键，如果有，则也删除/更新外键在子表中的记录。 |
| set null | 当在父表中删除对应记录时，首先检查该记录是否有对应外键，如果有则设置子表中该外键值为null（这就要求该外键允许取null）。 |
| set default | 父表有变更时，子表将外键列设置成一个默认的值 (Innodb不支持) |

Alter table表名 add constraint 外键名称 foreign key (外键字段) references主表名 (主表字段名) on update cascade on delete cascade;

多表查询

|  |  |  |
| --- | --- | --- |
| 内连接： |  | 连接查询的是两张表交集部分的数据 |

隐式内连接：select 字段列表 from 表1 , 表2 where 条件 ... ;

显示内连接：select字段列表 from表1 [inner] join 表2 on 连接条件 ... ;

表的别名:( 一旦为表起了别名，就不能再使用表名来指定对应的字段了，此时只能够使用别名来指定字段 )  
①. tablea as 别名1 , tableb as 别名2 ;  
②. tablea 别名1 , tableb 别名2 ;

|  |  |  |
| --- | --- | --- |
| 外连接： |  |  |

左外连接：select字段列表 from表1 left [ outer ] join表2 on 条件 ... ; 左外连接相当于查询表1(左表)的所有数据

右外连接：select字段列表 from表1 right [ outer ] join表2 on 条件 ... ; 右外连接相当于查询表2(右表)的所有数据

自连接：

自连接查询：select字段列表 from表A 别名A join 表A 别名B on 条件 ... ; 在自连接查询中，必须要为表起别名

联合查询：select字段列表 from表A ...  
union [ ALL ]  
select字段列表 from表B ....;

对于联合查询的多张表的列数必须保持一致，字段类型也需要保持一致。union all 会将全部的数据直接合并在一起，union 会对合并之后的数据去重。

子查询：(嵌套查询) 子查询外部的语句可以是insert / update / delete / selete 的任何一个

select \* fron t1 where column1 = ( select column1 from t2 );

标量子查询:（子查询结果为单个值）常用的操作符：= <> > >= < <=

查询 "销售部" 的所有员工信息

select \* from emp where dept\_id = (select id from dept where name = '销售部');

列子查询 : (子查询结果为一列) 常用的操作符：in 、not in 、 any 、some 、 all

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 例子 |
| in | 在指定的集合范围之内，多选一 | 查询 "销售部" 和 "市场部" 的所有员工信息  select \* from emp where dept\_id in (select id from dept where name = '销售部' or name = '市场部'); |
| not in | 不在指定的集合范围之内 |  |
| any | 子查询返回列表中，有任意一个满足即可 | 查询比研发部其中任意一人工资高的员工信息  select \* from emp where salary > any ( select salary from emp where dept\_id = (select id from dept where name = '研发部') ); |
| some | 与any等同，使用some的地方都可以使用any |  |
| all | 子查询返回列表的所有值都必须满足 | 查询比 财务部 所有人工资都高的员工信息  select \* from emp where salary > all ( select salary from emp where dept\_id = (select id from dept where name = '财务部') ); |

行子查询: (子查询结果为一行) 常用的操作符：= 、<> 、in 、not in

查询与 "张无忌" 的薪资及直属领导相同的员工信息 ;

select \* from emp where (salary,managerid) = (select salary, managerid from emp where name = '张无忌');

表子查询: (子查询结果为多行多列) 常用的操作符：in

查询与 "鹿杖客" , "宋远桥" 的职位和薪资相同的员工信息

select \* from emp where (job,salary) in ( select job, salary from emp where name = '鹿杖客' or name = '宋远桥' );

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 控制事务一 | | | 控制事务二 | | |
| 1）查看/设置事务提交方式 | 2）提交事务 | 3）回滚事务 | 1）开启事务 | 2）提交事务 | 3）回滚事务 |
| select @@autocommit ;  set @@autocommit = 0 ; | Commit; | Rollback; | start transaction 或 begin ; | Commit; | Rollback; |

事务四大特性（ACID）

原子性（Atomicity）：事务是不可分割的最小操作单元，要么全部成功，要么全部失败。

一致性（Consistency）：事务完成时，必须使所有的数据都保持一致状态。

隔离性（Isolation）：数据库系统提供的隔离机制，保证事务在不受外部并发操作影响的独立环境下运行。

持久性（Durability）：事务一旦提交或回滚，它对数据库中的数据的改变就是永久的。

并发事务问题

脏读：一个事务读到另外一个事务还没有提交的数据。

不可重复读：一个事务先后读取同一条记录，但两次读取的数据不同，称之为不可重复读。

幻读：一个事务按照条件查询数据时，没有对应的数据行，但是在插入数据时，又发现这行数据已经存在，好像出现了 "幻影"。

事务隔离级别（事务隔离级别越高，数据越安全，但是性能越低）

|  |  |  |  |
| --- | --- | --- | --- |
| 隔离级别 | 脏读 | 不可重复读 | 幻读 |
| Read uncommitted | √ | √ | √ |
| Read committed | × | √ | √ |
| Repeatable Read(默认) | × | × | √ |
| Serializable | × | × | × |

查看事务隔离级别：select @@transaction\_isolation;

设置事务隔离级别: set [ session | global ] transaction isolation level { read uncommitted | read committed | repeatable read | serializable }