# 函数

### 时间日期相关函数

|  |  |
| --- | --- |
| sysdate() | 获得当前时间 |
| ADDTIME (date2 ,time\_interval ) | 将time\_interval加到date2 |
| CURRENT\_DATE (  ) | 当前日期 |
| CURRENT\_TIME (  ) | 当前时间 |
| CURRENT\_TIMESTAMP ( ) | 当前时间戳 |
| DATE (datetime ) | 返回datetime的日期部分 |
| DATE\_ADD (date2 , INTERVAL d\_value d\_type ) | 在date2中加上日期或时间 |
| DATE\_SUB (date2 , INTERVAL d\_value d\_type ) | 在date2上减去一个时间 |
| DATEDIFF (date1 ,date2 ) | 两个日期差 |
| NOW (  ) | 当前时间 |
| YEAR|Month|DAY (datetime ) | 获得指定日期的年月日 |
| dayofmonth(datetime) | 获得指定日期的日 |

示例：

select addtime(‘02:30:30’,‘01:01:01’);注意：字符串、时间日期的引号问题

select date\_add(entry\_date,INTERVAL 2 year) from student;／／增加两年

select addtime(time,‘1 1-1 10:09:09’) from student;　／／时间戳上增加，注意年后没有-

|  |
| --- |
| -- dual,哑表,补全完整的sql语句.dual表并不存在  select sysdate() from dual  select now() from dual  select current\_date() from dual;  select current\_time() from dual;  select current\_timestamp() from dual;  select year('2021-01-01') from dual;  select year(sbirthday) from student;  select month('2021-01-01') from dual;  select month(sbirthday) from student;  select \* from student where month(sbirthday) = 5;  select day('2021-01-11') from dual  select dayofmonth('2021-01-11') from dual  select dayofmonth(sbirthday) from student  -- ADDTIME (date2 ,time\_interval ) 将time\_interval加到date2  select addtime('2021-01-01 00:00:00',30) from dual;-- 加30秒  select addtime('2021-01-01 00:00:00','01:59:59') from dual;  select addtime('2021-01-01 00:00:01','01:59:59') from dual;  -- DATE\_ADD (date2 , INTERVAL d\_value d\_type ) 在date2中加上日期或时间  -- 注意: 第二个参数位置,是三部分 1) 固定关键字 interval 2) 要加的日期值 3) 日期单位 day|month|year  select date\_add('2021-01-01',interval 32 day) from dual;  select date\_add('2021-01-01',interval 10 month) from dual;  select date\_add('2021-01-01',interval 10 year) from dual;  -- DATE\_SUB (date2 , INTERVAL d\_value d\_type ) 减  select date\_sub('2021-01-01',interval 1 day) from dual;  -- DATEDIFF (date1 ,date2 ) 两个日期差,date1-date2返回相差的天数,可以为负  select datediff('2021-01-01','2021-01-03') from dual; |

-- 获取本月生日的学生的个数 (1月)

select \* from student where date\_format(sbirthday,'%m') = date\_format(sysdate(),'%m')

select \* from student where month(now()) = month(sbirthday)

-- 获取近30天生日的学生

select sname,sbirthday from student

where date\_format(sbirthday,'%m%d')

between date\_format(now(),'%m%d')

and date\_format(date\_add(now(),interval 30 DAY),'%m%d')

### 字符串相关函数

|  |  |
| --- | --- |
| CHARSET(str) | 返回字串字符集 |
| CONCAT (str1[,... ]) | 连接字串 |
| INSTR (string ,substring) | 返回substring在string中出现的位置,没有返回0 |
| UCASE (string2 ) | 转换成大写 |
| LCASE (string2 ) | 转换成小写 |
| LEFT (string2 ,length ) | 从string2中的左边起取length个字符 |
| RIGHT (string2 ,length ) | 从string2中的右边起取length个字符 |
| LENGTH (string ) | string长度 |
| REPLACE (str ,search\_str ,replace\_str ) | 在str中用replace\_str替换search\_str |
| STRCMP (string1 ,string2 ) | 逐字符比较两字串大小, |
| SUBSTRING (str , position  [,length ]) | 从str的position开始,取length个字符 |
| LTRIM (string2 )  RTRIM (string2 )  trim | 去除前端空格或后端空格 |

示例:

SELECT CHARSET('QIUSHIJU');

|  |
| --- |
| select charset(sname) from stu;  -- CONCAT (str1[,... ]) 连接字串  select concat('a','b') from dual;  select concat('我叫',sname) from stu;  -- INSTR (string ,substring ) 返回substring在string中出现的位置,没有返回0  -- 即判断是否包含字符串  select instr('java','av') from dual;  select instr('java','aa') from dual;  select instr(sname,'三') from stu;  select if(instr(sname,'三') = 0,'不包含三','包含三' ) from stu;  -- UCASE (string2 ) 转换成大写  select ucase('smith') from dual;  select ucase('爪哇') from dual;  -- LEFT (string2 ,length ) 从string2中的左边起取length个字符  select left('java',2) from dual  -- 查询stu表中不同姓氏的人数  select count(distinct left(sname,1) )from stu  -- RIGHT (string2 ,length ) 从string2中的右边起取length个字符  select right('java',2) from dual;  -- LENGTH (string ) string长度,中文在utf8编码占3个,英文占1个  select length(sname) from stu  select length('a') from dual  -- REPLACE (str ,search\_str ,replace\_str ) 在str中用replace\_str替换search\_str  select replace('java','av','AV') from dual  select replace(sname,'三','叁') from stu  -- STRCMP (str1 ,str2 ) 逐字符比较两字串大小,  -- str1 < str2 返回 -1  -- str1 > str2 返回 1  -- str1 = str2 返回 0  -- ===> 参考Java.TreeSet集合的排序算法  select strcmp('a','b') from dual  select strcmp('b','a') from dual  select strcmp('b','b') from dual  -- SUBSTRING (str , position [,length ]) 从str的position开始,取length个字符  -- position是位置,不是下标,即第一个字符就是1  select substring('javabigdata',5) from dual  select substring('javabigdata',5,3) from dual  -- 滤空  select ltrim(' java ') from dual;  select rtrim(' java ') from dual;  select trim(' java ') from dual; |

### 字符-日期互转

|  |
| --- |
| date\_format(date,'%Y-%m-%d') -------------->oracle中的to\_char();  　　str\_to\_date(date,'%Y-%m-%d') -------------->oracle中的to\_date();  　　%Y：代表4位的年份  　　%y：代表2为的年份  　　%m：代表月, 格式为(01……12)  　　%c：代表月, 格式为(1……12)  　　%d：代表月份中的天数,格式为(00……31)  　　%e：代表月份中的天数, 格式为(0……31)  　　%H：代表小时,格式为(00……23)  　　%k：代表 小时,格式为(0……23)  　　%h： 代表小时,格式为(01……12)  　　%I： 代表小时,格式为(01……12)  　　%l ：代表小时,格式为(1……12)  　　%i： 代表分钟, 格式为(00……59)  　　%r：代表 时间,格式为12 小时(hh:mm:ss [AP]M)  　　%T：代表 时间,格式为24 小时(hh:mm:ss)  　　%S：代表 秒,格式为(00……59)  　　%s：代表 秒,格式为(00……59)  　　SELECT DATE\_FORMAT(20130111191640,'%Y-%m-%d %H:%i:%s')  　　DATE\_FORMAT(20130111191640,'%Y-%m-%d %H:%i:%s') |

|  |
| --- |
| -- 日期到字符串  select date\_format(now(),'%Y年%m月%d日') from dual  -- 字符串到日期  select str\_to\_date('2021年01月11日','%Y年%m月%d日') from dual;  -- 错误 insert into student (sid,sname,sbirthday) values (21,'猪八戒','2021年01月01日')  -- 正确 insert into student (sid,sname,sbirthday) values (21,'猪八戒','2021-01-01')  insert into student (sid,sname,sbirthday) values (21,'猪八戒',str\_to\_date('2021年01月01日','%Y年%m月%d日')) |

### 数学相关函数

|  |  |
| --- | --- |
| ABS (number2) | 绝对值 |
| BIN (decimal\_number ) | 十进制转二进制 |
| CEILING (number2 ) | 向上取整 |
| CONV(number2,from\_base,to\_base) | 进制转换 |
| FLOOR (number2 ) | 向下取整 |
| FORMAT (number,decimal\_places ) | 保留小数位数 |
| HEX (DecimalNumber ) | 转十六进制 |
| LEAST (number , number2  [,..]) | 求最小值 |
| MOD (numerator ,denominator ) | 求余 |
| RAND([seed]) | RAND([seed]) |
| round(x[,d]) | 四舍五入 |

示例:

SELECT ABS(-101);

|  |
| --- |
| -- abs(x) 取绝对值  select abs(1) from dual  select abs(-1) from dual  -- CEILING (number2 ) 向上取整  select ceiling(11.1) from dual;  -- FLOOR (number2 ) 向下取整  select floor(11.1) from dual;  select floor(-11.1) from dual;  -- FORMAT (number,decimal\_places ) 保留小数位数,保留时会四舍五入  select format(99.12645,2) from dual;  select format(score,2) from stu  select format(score,0) from stu  -- 随机数,产生0-1之间随机数  select rand() from dual;  -- 种子,会根据种子计算出一个随机数.但多次运行种子一样,结果一样  select rand(1) from dual;  -- round(x) 四舍五入  select round(1.1) from dual;  select round(1.5) from dual;  -- round(x,d) 四舍五入,保留d位小数  select round(1.1234,2) from dual;  select round(1.1254,2) from dual;  -- 使用mysql函数,产生0-10之间的随机数  select round(rand()\*10) from dual |

事务transaction – TX

需求:有一张银行账户表,有a用户给b用户转账.A账户钱减少,b账户钱增加.但是当a账户钱减少时,断电了,B账户并没有增加钱.

解决方案:a减少钱,但是不要立即修改数据表,b的钱增加之后,同时修改表.

事务：

事务指逻辑上的一组操作，组成这组操作的各个单元，要么全部成功，要么全部不成功。

演示:

CREATE TABLE `account` (

`id` int(50) NOT NULL,

`name` varchar(50) NOT NULL,

`money` int(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO account VALUES(1,'张三',1000);

INSERT INTO account VALUES(2,'李四',1000);

修改a的钱

Update account set money=money-1000 where id=1;

此时退出mysql,再次进去查看b账户的钱.没有增加

# １、Mysql中的事务

a、mysql引擎是支持事务的

b、mysql默认自动提交事务。每条语句都处在单独的事务中。

-- 查看当前数据库事务的提交方式

SHOW VARIABLES; [like ‘autocommit’]

* 查看当前数据库事务的提交方式

select @@autocommit; 0是手动提交,1是自动提交

* 设置当前数据库的提交方式

set @@autocommit = 0

看到是autocommit on 开启自动提交

手动关闭: set autocommit = off / 或者= 0

再演示:直接修改数据,在别的窗口可以查看效果,数据没有修改.需要手动处理:提交或回滚

c、手动控制事务

开启事务：start transaction | begin

提交事务：commit

回滚事务：rollback

手动回滚:

BEGIN; -- 开启事务

UPDATE account SET money=money-100 WHERE `name`='张三';

ROLLBACK; -- 回滚,数据不会变动

|  |
| --- |
| -- =================== 原始数据 ======================  CREATE TABLE `account` (  `id` int(50) NOT NULL,  `name` varchar(50) NOT NULL,  `money` int(255) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  INSERT INTO account VALUES(1,'张三',1000);  INSERT INTO account VALUES(2,'李四',1000);  -- ================== 默认事务 ==================  /\*  mysql默认每句sql都是处于事务,自动开启,提交.  \*/  -- 查看当前提交方式  SHOW VARIABLES like 'autocommit'  -- A向B转账100  -- A减少100  update account set money = money - 100 where id = 1;  -- B增加100  update account set money = money + 100 where id = 2;  -- =============== 手动控制事务 ================  -- 开启事务  begin;  -- A减少100  update account set money = money - 100 where id = 1;  -- 假如此处出现事故,,下个语句不执行,可以执行rollback回滚数据  -- B增加100  update account set money = money + 100 where id = 2;  -- 提交事务  commit;  -- 出错,可以回滚  rollback; |

# 2、事务的特性（ACID）（面试题）

**原子性**（Atomicity）：指事务的整个操作是一个整体，要么都成功,要么都失败

**一致性**（Consistency）：事务必须使数据库从一个一致性状态变换到另外一个一致性状态。转账前和转账后的总金额不变。

**隔离性**（Isolation）：事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。

* 讲解:一个事务A开始事务,修改a表数据,自己查询数据,已经修改
* 另外开启一个事务B:操作同一个表-> 修改a表其他列数据,自己查看已经修改
* 但是A事务再查表数据,并未看见B事务做出的修改.同样B事务也看不见A事务做出的修改.

**持久性**（Durability）：指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响。

# 3、事务的隔离级别

隔离级别:

数据库通过设置事务的**隔离级别**防止以上情况的发生：

1、读未提交-READ UNCOMMITTED: 赃读、不可重复读、虚读都有可能发生。

2、读已提交-READ COMMITTED: 避免赃读。不可重复读、虚读都有可能发生。（oracle默认的）

3、可重复读-REPEATABLE READ:避免赃读、不可重复读。虚读有可能发生。（**mysql默认**）

4、串行化-SERIALIZABLE: 避免赃读、不可重复读、虚读。

不同的隔离级别产生如下情况:

**赃读**：指一个事务读取了另一个事务未提交的数据。

**不可重复读**：在一个事务内读取表中的某一行数据，多次读取结果不同。一个事务读取到了另一个事务提交后的数据。（update）

**虚读**（幻读）：是指在一个事务内读取到了别的事务插入的数据，导致前后读取不一致。 (insert)

|  |
| --- |
| 脏读:  对于两个事物 T1, T2, T1 读取了已经被 T2 更新但还没有被提交的字段.  之后, 若 T2 回滚, T1读取的内容就是临时且无效的. |
| 不可重复读:  对于两个事物 T1, T2, T1 读取了一个字段, 然后 T2 更新了该字段.  之后, T1再次读取同一个字段, 值就不同了. |
| 幻读:  对于两个事物 T1, T2, T1 从一个表中读取了一个字段, 然后 T2 在该表中  插入了一些新的行. 之后, 如果 T1 再次读取同一个表, 就会多出几行. |

以上四种隔离级别最高的是Serializable级别，最低的是Read uncommitted级别，当然级别越高，执行效率就越低。像Serializable这样的级别，就是以锁表的方式(类似于Java多线程中的锁)使得其他的线程只能在锁外等待，所以平时选用何种隔离级别应该根据实际情况。在MySQL数据库中默认的隔离级别为Repeatable read (可重复读)。

mysql中：

查看当前的事务隔离级别：SELECT @@TX\_ISOLATION;

更改当前的事务隔离级别：

SET [glogal | session] TRANSACTION ISOLATION LEVEL 四个级别之一。

设置隔离级别必须在事务之前

案例演示：

Navicat演示隔离级别效果。

|  |
| --- |
| -- ================== 默认事务 ==================  /\*  mysql默认每句sql都是处于事务,自动开启,提交.  \*/  -- 查看当前提交方式  SHOW VARIABLES like 'autocommit'  -- A向B转账100  -- A减少100  update account set money = money - 100 where id = 1;  -- B增加100  update account set money = money + 100 where id = 2;  -- =============== 手动控制事务 ================  -- 开启事务  begin;  -- A减少100  update account set money = money - 100 where id = 1;  -- 假如此处出现事故,,下个语句不执行,可以执行rollback回滚数据  -- B增加100  update account set money = money + 100 where id = 2;  -- 提交事务  commit;  -- 出错,可以回滚  rollback;  -- ====================== 隔离级别,以及对应的bug ================  -- 查看当前的事务隔离级别：  SELECT @@TX\_ISOLATION;  -- 修改隔离级别  SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ  -- ======> 一,设置级别为读未提交  SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  -- ======> 二,设置级别为读已提交  SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;  -- ======> 三,设置级别为可重复读  SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ;  -- ======> 三,设置级别为可重复读  SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;  -- 开事务  begin;  -- 改数据  update account set money = money - 100 where id =1;  INSERT INTO account VALUES(4,'赵六',1000);  select \* from account where id = 1;  commit; |

# 锁机制

Innodb默认是行锁,但是在事务(增删改)中,没有使用到索引,那么系统会自动升级会表锁,

行锁:事务开始后,其他用户不能操作该行数据,

表锁:事务开始后,其他用户不能操作这张表