# SQL[sql中的变量使用]

|  |
| --- |
| SELECT DATE\_ADD('2018-01-01',INTERVAL @i := @i + 1 DAY) AS DATE  FROM  (SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1 UNION ALL SELECT 1) AS tmp, (SELECT @i := -1) t; |

# 

# SQL[sql中的判断]

CASE WHEN THEN...

# SQL[乐观锁]

乐观锁就是执行的时候不会拿到锁资源，当真正要修改的时候才会去查询数据版本，在进行更新。

数据库：

Id name price vsersion

Mysql INNOB 默认在delete,update,insert 自动添加排加锁。不会存在同时进行修改，只有先后顺序

# SQL[悲观锁]

begin;

SELECT \* from city where id = "1" lock in share mode

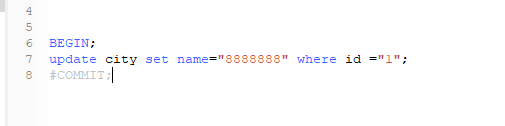
COMMIT;

上面就是开启了事物，lock in share mode（共享锁）

只有当commit的时候其他的才能进行操作。

注意：【mysql innob 机制默认给insert,delete,update添加勒for update 加排锁】

一条数据不能同时添加共享锁和加排锁。



加排锁，只要没有commit就不会更新事物，同时锁住这排数据

# SQL[生成log文件]

SHOW VARIABLES LIKE "general\_log%"

SET GLOBAL general\_log = 'ON;

SET GLOBAL general\_log\_file='D:\\sql.log';zl;。

# SQL[间隔查询数据]

业务：统计报表的时候，查询间隔数据。

假设我要查询出 从今天开始，查过去七天的数据。

|  |
| --- |
| SELECT  SUM(log.OrderTotalMoney) as IndexDateValue, DATE\_FORMAT(DATE\_SUB(NOW(),INTERVAL interval1-1 DAY) , '%Y-%m-%d';) as IndexDate  FROM  (SELECT @dayinsertval:=@dayinsertval+1 as interval1 FROM mysql.help\_topic A,(SELECT @dayinsertval :=0 days) B WHERE help\_topic\_id < 6) inter  LEFT JOIN  orderlog log  ON  DATE\_FORMAT(log.OrderCreateTime, &apos;%Y-%m-%d &apos;) = DATE\_FORMAT(DATE\_SUB(NOW(),INTERVAL interval1-1 DAY), &apos;%Y-%m-%d &apos;)  GROUP BY interval1 ORDER BY interval1  ; |

什么思想呢？

首先：

SELECT @dayinsertval:=@dayinsertval+1 as interval1 FROM mysql.help\_topic A,(SELECT @dayinsertval :=0 days) B WHERE help\_topic\_id < 6

Mysql.help\_topic 是一个表，id是从1开始自增的一张表。 这里就是让@dayinsertval变量进行循环自增。

第二部：

DATE\_FORMAT(log.OrderCreateTime, &apos;%Y-%m-%d &apos;) = DATE\_FORMAT(DATE\_SUB(NOW(),INTERVAL interval1-1 DAY), &apos;%Y-%m-%d &apos;)

从今天开始，减去interval变量-1就等于说是从今天开始，然后进行比较。

第三部：

Group by分组。因为很有可能一天没有数据。但是也要查询出来给后台使用，所以我们要保留一个位置，所以我们直接将interval 进行分组。

核心就是：虚拟一张表，定义一个变量，不断的给interval变量进行递增，然后进行判断。

# SQL[存储过程字段名封装]

|  |
| --- |
| -- 字段  select group\_concat(COLUMN\_NAME) COLUMN\_NAME from information\_schema.COLUMNS where table\_name = 'memrecharge' INTO @fieldString;  -- 生成变量 into @asdasda  select group\_concat(CONCAT('@',COLUMN\_NAME)) COLUMN\_NAME from information\_schema.COLUMNS where table\_name = 'memrecharge' INTO @intofieldString;  -- 拼接SQL、 SELECT XXXXX INTO XXXXX from FROM memrecharge WHERE RechargeID = 'xxxxxxx';  SET @sqlString=CONCAT('SELECT ',CONCAT(@fieldString,CONCAT(' into ',@intofieldString),' FROM memrecharge WHERE RechargeID ='),CONCAT("'",memrechargeid,"'"));  SELECT @sqlString;  PREPARE stmt FROM @sqlString; -- 预处理动态sql语句  EXECUTE stmt; -- 执行sql语句  deallocate prepare stmt; -- 释放prepare |

这里运用的就是information\_schema.COLUMNS的表查询出了所有的表字段名，然后group\_concat函数自动拼接的字段名。

然后进行拼接SQL

这里还有个技术就是：预处理动态sql语句

PREPARE stmt FROM @sqlString; -- 预处理动态sql语句

EXECUTE stmt; -- 执行sql语句

deallocate prepare stmt; -- 释放prepare