



MySQL数据库开发技术

—— 数据操作与事务控制

本章内容

节	知识点	掌握程度	难易程度
数据操作语言	数据操作语言	理解	
插入数据	插入数据	理解	
	INSERT语法结构	掌握	
	插入空值	掌握	
	插入日期值	掌握	
	插入多行数据	掌握	
修改数据	修改数据	理解	
	UPDATE语法结构	掌握	
	修改部分行记录	掌握	
	修改所有行记录	掌握	
	修改多列	掌握	
删除数据	删除数据	理解	
	语法结构	掌握	
	删除选中记录	掌握	
	删除全部记录	掌握	
事务	事务概念及特征	理解	难
	事务处理	理解	难

数据操作语言

- 数据操作语言

- Data Manipulation Language，简称DML，主要用来实现对数据库表中的数据进行操作。
- 数据操作语言主要包括如下几种：
 - 增加行数据：使用INSERT语句实现
 - 修改行数据：使用UPDATE语句实现
 - 删除行数据：使用DELETE语句实现


插入数据

50	DEVELOPMENT	DETROIT
----	-------------	---------

New row

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

“...insert a new row
into DEPT table...”



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

插入数据

- 使用 INSERT 语句向表中插入数据，语法结构如下：

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- 采用这种语法一次只能追加一条记录；
- column 部分叫做列名列表，value 部分叫做值列表，列名列表和值列表必须在个数及数据类型上保持一致；
- 列名列表部分可以省略，如果省略，默认包括该表的所有列，列的顺序为使用 desc 表名 命令所查看的顺序；
- 列名列表部分也可以指定部分非空的列，注意值列表必须和列名列表对应；
- 字符和日期型数据必须要用单引号括起来。

```
SQL> INSERT INTO      dept (deptno, dname, loc)  
      2  VALUES      (50, 'DEVELOPMENT', 'DETROIT');  
1 row created.
```

插入数据

- 插入空值NULL

- 隐含法：在列名列表中忽略该列。

```
SQL> INSERT INTO      dept (deptno, dname )  
      2 VALUES        (60, 'MIS') ;  
1 row created.
```

- 显示法：指定 NULL关键字或者''。

```
SQL> INSERT INTO      dept  
      2 VALUES        (70, 'FINANCE', NULL) ;  
1 row created.
```

练习1

- 1. 向部门表新增一个部门，部门编号为50，部门名称为HR，工作地点为SY。
- 2. 向部门表新增一个部门，部门编号为60，部门名称为MARKET。

插入数据

- 插入日期值
 - SYSDATE() 函数记录当前日期和时间

```
SQL> INSERT INTO      emp (empno, ename, job,  
2                      mgr, hiredate, sal, comm,  
3                      deptno)  
4  VALUES             (7196, 'GREEN', 'SALESMAN',  
5                      7782, SYSDATE(), 2000, NULL,  
6                      10);  
1 row created.
```


插入数据

- 插入日期值

```
SQL> INSERT INTO emp
2  VALUES      (2296, 'AROMANO', 'SALESMAN', 7782,
3                '1997-02-03',
4                1300, NULL, 10);
1 row created.
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
2296	AROMANO	SALESMAN	7782	1997-02-03	1300		10

练习2

- 1. 向员工表中新增一个员工，员工编号为8888，姓名为BOB，岗位为CLERK，经理为号7788，入职日期为1985-03-03，薪资3000，奖金和部门为空。

批量插入数据

- 使用insert语句可以一次性地向表批量插入多条记录，语法格式如下。

```
INSERT INTO 表名[(字段列表)] VALUES  
(值列表1),  
(值列表2),  
...  
(值列表n);
```

批量插入数据

```
INSERT INTO EMP(EMPNO, ENAME, JOB, SAL) VALUES  
( '8881', '张三', '部门经理', 6000),  
( '8882', '李四', '职员', 3000),  
( '8883', '王五', '职员', 3500),  
( '8884', '赵六', '部门经理', 6500),  
( '8885', '高七', '职员', 2500),  
( '8886', '马八', '职员', 3100),  
( '8887', '钱九', '部门经理', 5000),  
( '8888', '孙十', '职员', 2800);
```

批量插入数据

- 通过子查询插入多行数据
 - 语法

```
INSERT INTO 表名[(列名1[,列名2, ..., 列名n])] 子查询 ;
```

- 不必书写values子句
- INSERT子句和数据类型必须和子查询中列的数量和类型相匹配中列的数量

批量插入数据

- 创建manager表

```
SQL> CREATE TABLE manager AS  
2 SELECT * FROM emp WHERE 1=0;
```

- 向manager表中插入职位为MANAGER的记录

```
SQL> INSERT INTO manager  
2 SELECT *  
3 FROM emp  
4 WHERE job = 'MANAGER';  
3 rows created.
```

练习3

- 1. 使用CREATE TABLE emp_back as
SELECT * FROM EMP WHERE 1=0, 创建
emp_back表, 拷贝下来即可。
- 2. 把emp表中入职日期大于1982年1月1日之前的员工信息复制到emp_back表中。

修改数据

- 修改数据主要用来按照指定条件修改表中某些行的列数据。

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				



EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				20

修改数据

- 修改数据使用UPDATE子句完成，语法结构如下：

```
UPDATE      table  
SET         column = value [, column = value]  
[WHERE      condition];
```

- WHERE子句用来限定修改哪些行。
- SET子句用来限定修改哪些列。
- WHERE子句中的更新条件是一个逻辑表达式，通常需要使用到关系运算符和逻辑运算符，返回True或者False。

修改数据

关系运算符和比较函数	功能描述
=	等于，例如a=1
<=>	与=相同，但如果操作符两边的操作数都是NULL，则表达式返回1，而不是NULL；而当只有一个操作数为NULL时，其所得值为0而不为NULL
!=	不等于，例如a!=1
<>	与!=相同，例如a<>1
<=	小于或等于，例如a<=1
<	小于，例如a<1
>=	大于或等于，例如a>=1
>	大于，例如a>1
IS NULL	判断指定的值是否为NULL，例如a IS NULL
IS NOT NULL	判断指定的值是否不为NULL，例如a IS NOT NULL
BETWEEN...AND	判断操作数是否在指定的范围之间，例如a BETWEEN 1 AND 100
NOT BETWEEN...AND	判断操作数是否不在指定的范围之间，例如a NOT BETWEEN 1 AND 100
COALESCE	返回列表中第一个非NULL的值，如果没有非NULL值，则返回NULL。例如COALESCE(NULL, NULL, 1, 2)的结果为1
GREATEST	当参数列表中有两个或多个值，则返回其中最大的值。例如，GREATEST(1,2,3)的结果为3
IN	判断表达式是否为列表中的一个值，例如a IN (1,2,3,4)，如果a为1,2,3或4，则表达式返回True，否则返回False
NOT IN	判断表达式是否为列表中的一个值，例如a NOT IN (1,2,3,4)，如果a为1,2,3或4，则表达式返回False，否则返回True
ISNULL(expr)	判断指定的表达式是否为NULL
LEAST	当参数列表中有两个或多个值，则返回其中最小的值。例如，LEAST (1,2,3)的结果为1

修改数据

关系运算符	功能描述
NOT	逻辑非。当操作数为0时结果为1；当操作数为1时结果为0
!	与NOT相同
AND	逻辑与。例如a AND b，如果a和b都等于True时，则返回True，否则返回False
&&	与AND相同
OR	逻辑非。例如a OR b，如果a和b中有一个等于True时，返回True，否则返回False
	与OR相同
XOR	逻辑异或。例如a XOR b 的计算等同于(a AND (NOT b)) OR ((NOT a) AND b)

修改数据

- 使用 WHERE 子句指定要修改的记录
 - 把员工编号为7782的部门编号修改为20

```
SQL> UPDATE emp
2 SET deptno = 20
3 WHERE empno = 7782;
1 row updated.
```

- 如果要修改所有记录，WHERE子句可以忽略
 - 把所有员工的部门编号修改为20

```
SQL> UPDATE emp
2 SET deptno = 20;
14 rows updated.
```

修改数据

- 一次修改多列

- 把部门编号为10的员工，部门编号调整为20，工资增加100

```
SQL> UPDATE    emp
      2  SET      deptno = 20,sal=sal+100
      3  WHERE    deptno = 10;
5 row updated.
```

修改数据

- 修改记录时的完整性约束错误
 - 把部门编号为10的员工，部门编号调整为55

```
SQL> UPDATE emp
      2 SET deptno = 55
      3 WHERE deptno = 10;
```

UPDATE emp

*

ERROR 位于第 1 行:

ORA-02291: 违反完整约束条件 (SCOTT.FK_DEPTNO) - 未找到父项关键字


练习4

- 1. 修改部门20的员工信息，把82年之后入职的员工入职日期向后调整10天
- 2. 修改奖金为null的员工，奖金设置为0
- 3. 修改工作地点在NEW YORK或CHICAGO的员工工资，工资增加500

删除数据

- 删除数据主要用来按照指定条件从表中删除某些行。

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

删除数据

- 语法结构
 - 使用 DELETE 语句删除表中满足条件的行记录

```
DELETE FROM      table  
[WHERE           condition];
```

删除数据

- 删除选中记录
 - 删除职位是CLERK的员工记录

```
SQL> DELETE FROM emp
      2  WHERE          job = 'CLERK';
已删除4行
```

- 删除全部记录
 - 删除所有员工记录

```
SQL> DELETE FROM emp;
14 rows deleted.
```

删除数据

- 删除记录时的完整性约束错误
 - 删除部门编号为10的部门记录

不能删除该数

```
SQL> DELETE FROM dept
      2 WHERE deptno = 10;
```

DELETE FROM dept

*

ERROR 位于第 1 行:

ORA-02292: 违反完整约束条件 (SCOTT.FK_DEPTNO) - 已找到子
记录日志

如果数据行。

删除数据

- 截断表
 - 截断表语法:

```
TRUNCATE TABLE table;
```

```
TRUNCATE TABLE emp;
```

- TRUNCATE和DELETE区别
 - TRUNCATE是DDL，只能删除表中所有记录，释放存储空间，使用ROLLBACK不可以回滚。
 - DELETE是DML，可以删除指定记录，不释放存储空间，使用ROLLBACK可以回滚。

练习5

- 1. 删除经理编号为7566的员工记录
- 2. 删除工作在NEW YORK的员工记录
- 3. 删除工资大于所在部门平均工资的员工记录

事务处理

- 事务(Transaction)概念

- 事务：也称工作单元，是由一个或多个SQL语句所组成的操作序列，这些SQL语句作为一个完整的工作单元，要么全部执行成功，要么全部执行失败。在数据库中，通过事务来保证数据的一致性。
- 事务处理语言：Transaction Process Language，简称TPL，主要用来对组成事务的DML语句的操作结果进行确认或取消。确认也就是使DML操作生效，使用提交(COMMIT)命令实现；取消也就是使DML操作失效，使用回滚(ROLLBACK)命令实现。
- 通过事务的使用，能防止数据库中出现数据不一致现象。如两个银行账户进行转账，涉及到两条更新操作，这两条更新操作只允许全部成功或失败，否则数据会出现不一致的现象。

事务处理

- 事务组成

- 在数据库中，事务由一组相关的DML或SELECT语句，加上一个TPL语句（COMMIT、ROLLBACK）**或**一个DDL语句（CREATE、ALTER、DROP、TRUNCATE等）**或**一个DCL（GRANT、REVOKE）语句。
- 例：如下语句组成两个事务。
 - INSERT...
 - UPDATE...
 - DELETE...
 - SELECT...
 - INSERT...
 - COMMIT; -- 前6条语句，组成第1个事务
 - UPDATE...
 - DELETE...
 - CREATE... ; -- 后3条语句，组成第2个事务

事务处理

- 事务特征可用四个字母的缩写表示：即ACID
 - 原子性 (Atomicity)
 - 事务就像“原子”一样，不可被分割，组成事务的DML操作语句要么全成功，要么全失败，不可能出现部分成功部分失败的情况。
 - 一致性 (Consistency)
 - 一旦事务完成，不管是成功的，还是失败的，整个系统处于数据一致的状态。
 - 隔离性 (Isolation)
 - 一个事务的执行不会被另一个事务所干扰。比如两个人同时从一个账户取钱，通过事务的隔离性确保账户余额的正确性。
 - 持久性 (Durability)
 - 也称为永久性，指事务一旦提交，对数据的改变就是永久的，不可以再被回滚。

事务处理

- MySQL的事务处理主要有两种方法
 - 用begin, rollback, commit来实现
begin开始一个事务
rollback事务回滚
commit 事务提交
 - 直接用set来改变MySQL的自动提交模式
MySQL默认是自动提交的，也就是你提交一个sql，就直接执行！可以通过
set autocommit = 0 禁止自动提交
set autocommit = 1 开启自动提交
来实现事务的处理。
- 但要注意当用set autocommit = 0 的时候，以后所有的sql都将作为事务处理，直到用commit确认或 rollback结束，注意当结束这个事务的同时也开启了新的事务！按第一种方法只将当前的做为一个事务！

事务处理

- 隐式结束

- 隐式提交：当下列任意一种情况发生时，会发生隐式提交
 - 执行一个DDL (CREATE、ALTER、DROP、TRUNCATE、RENAME) 语句；
 - 执行一个DCL (GRANT、REVOKE) 语句；
- 隐式回滚：当下列任意一种情况发生时，会发生隐式回滚
 - 客户端强行退出
 - 客户端连接到服务器端异常中断
 - 系统崩溃

事务处理

- 设置保存点：如果在一个事务内，想要回滚到指定位置，不是回滚到事务的起始点，可以通过保存点(SAVEPOINT)来实现。
 - `SAVEPOINT savepointname;` --定义一个保存点语句；
 - `ROLLBACK TO savepointname;` --回滚到指定保存点
 - 注意：如上两条语句不结束事务的执行。

事务处理

- 例: 分析如下操作序列

BEGIN;	——开始事务操作
DELETE FROM test ;	
ROLLBACK;	——撤消DELETE操作
INSERT INTO test VALUES ('A');	
SAVEPOINT insert_a;	——定义insert_a保存点
INSERT INTO test VALUES ('B');	
SAVEPOINT insert_b;	——定义insert_b保存点
INSERT INTO test VALUES ('C');	
ROLLBACK TO insert_b;	——撤消操作到insert_b保存点
DELETE FROM test WHERE test_str = 'A';	
COMMIT;	——将所有修改写入数据库
ROLLBACK;	——所有操作已经COMMIT提交, 不能回滚

练习5

- 1. test表为空表，分析如下语句操作后，最后test表的状态。
 - BEGIN
 - INSERT INTO test(id,name) values(1, 'a');
 - INSERT INTO test(id,name) values(2, 'b');
 - SAVEPOINT s1;
 - INSERT INTO test(id,name) values(3, 'c');
 - INSERT INTO test(id,name) values(4, 'd');
 - DELETE FROM test WHERE id in (1,3);
 - ROLLBACK TO s1;
 - DELETE FROM test WHERE id in (2,4);
 - COMMIT;
 - ROLLBACK;

小结

- 插入数据的方法：一次插入一行，插入空值、插入特殊值、插入多行；
- 修改表中数据：修改指定行，修改所有列、修改多列；
- 删除表中数据：删除指定行、删除所有行；
- 数据库事务：事务组成、事务特性、事务处理；

课后作业

1. 使用如下语句，创建学生表student和班级表class

```
create table student (           --学生表
    xh char(4), --学号
    xm varchar(10), --姓名
    sex char(2), --性别
    birthday date, --出生日期
    sal double(7, 2), --奖学金
    studentcid int(2) --学生班级号
)
Create table class (    --班级表
    classid int(2), --班级编号
    cname varchar(20), --班级名称
    ccount int(3) --班级人数
)
```

课后作业

2. 基于上述学生表和班级表，完成如下问题

(1) 添加三个班级信息为：1, JAVA1班, null

2, JAVA2班, null

3, JAVA3班, null

(2) 添加学生信息如下： 'A001', '张三', '男', '01-5月-05', 100, 1

(3) 添加学生信息如下： 'A002', 'MIKE', '男', '1905-05-06', 10

(4) 插入部分学生信息： 'A003', 'JOHN', '女'

(5) 将A001学生性别修改为'女'

(6) 将A001学生信息修改如下：性别为男，生日设置为1980-04-01

(7) 将生日为空的学生班级修改为Java3班

(8) 请使用一条SQL语句，使用子查询，更新班级表中每个班级的人数字段

课后作业

3. 使用如下语句，建立以下表

```
CREATE TABLE copy_emp (  
    empno int(4),  
    ename varchar(20),  
    hiredate date,  
    deptno int(2),  
    sal double(8, 2))
```

课后作业

4. 在第三题表的基础上，完成下列问题

- (1) 在表copy_emp中插入数据，要求sal字段插入空值，部门号50，参加工作时间为2000年1月1日，其他字段随意
- (2) 在表copy_emp中插入数据，要求把emp表中部门号为10号部门的员工信息插入
- (3) 修改copy_emp表中数据，要求10号部门所有员工涨20%的工资
- (4) 修改copy_emp表中sal为空的记录，工资修改为平均工资
- (5) 把工资为平均工资的员工，工资修改为空
- (6) 另外打开窗口2查看以上修改
- (7) 执行commit，窗口2中再次查看以上信息
- (8) 删除工资为空的员工信息
- (9) 执行rollback

Neuedu

