MySql

一　数据库概念

1　什么是数据库

数据库（database）是按照数据结构组织、存储和管理数据的仓库，简而言之就是存储数据的仓库

2　数据库分类

发展历程：层次式数据库，网络式数据库，关系型数据库
关系型数据库：利用表来存储数据，利用表和表之间的关系保存数据之间的关系

3　常见的关系型数据库

SQLserver
mysql
oracle
...
任何关系型数据库的语法都是 sql

4　mysql 介绍

mysql 是一个关系型数据库，由瑞典的 mysql AB 公司开发，目前属于 oracle 旗下
特点是*体积小，速度快，成本低，开源*

5　mysql 安装配置

详见安装文档

6　mysql 常见概念

数据库：一个 mysql 服务器中可以有多个库，一个库就是表的集合

表：类似于一个 excel 表格，用于存储数据

表记录：表中的一行数据

二　SQL 分类

1 DQL

2 DML

3 DDL

4 DCL

5 TCL

三 Mysql 数据库基本命令

1 登录

1.A mysql –u 用户名 –P *密码* –h *ip*

1.B 例：mysql –uroot –p1234 –h 127.0.0.1

1.C 注意：

1.C.1 红色部分不可省略，灰色部分”*密码* –h *ip*”可省略，如果是本机，ip 可以省略

1.C.2 –u *空格*用户名 –h *空格* ip 中间的*空格*可以省略也可不省略

1.C.3 -p 后面可以不写密码，换行继续写，*安全

2 退出

2.A.1 ***quit;***

2.A.2 ***exit;***

2.A.3 ***快捷键 Ctrl+C***

3 启动/关闭服务

3.A.1 *net stop mysql*

3.A.2 *net start mysql*

3.A.3 mysql 为服务名

四 SQL 语句

注释：除了 SQL 标准之外，大部分 SQL 数据库程序都拥有它们自己的专有扩展！

1 注意

1.A 结束有英文分号

1.B Query OK, 1 row affected (0.00 sec)：操作成功，一行被改变

1.C 不区分大小写(命令不区分，数据库名区分,字段等)

2 库的 CRUD

2.A.1 *Create ： create database db_name*

2.A.2 *Read :show databases;*

2.A.3 *Update :alter database db_name character set gbk;*

2.A.4 *Delete:drop database db_name;*

3 表的 CRUD

4 表记录的 CRUD

5 数据类型

5.A 字符串

5.A.1 char(n) *定长字符串*

n 表示字符串的最大长度不能超过这个值，n 最大为 255
存储时分配固定长度的空间，一般用于存储固定长度的字符串，如手机号、身份证号等
效率高，相对空间浪费
Sqlser 中多余空间空格补充，而 mysql 中不会

5.A.2 varchar(n) *不定长字符串*

n 最大为 65535，n 只是规定最长字符串
存储时，先计算字符串的大小，然后再分配对应长度的空间，一般存储不固定长度的字符，如姓名、昵称等，节省空间

5.B 数值类型

数据库	Java(类似)
tinyint	byte
smallint	short
int	int
bigint	long
float	float
double	double

5.C 大数据类型

5.C.1 blob：大二进制类型，可以存储二进制数据，如图片、音频、视频，最大 4gb

5.C.2 text：大文本类型，可以存储大量字符数据，最大 5gb

text 为 mysql 方言，其他数据库中为 clob

5.D 日期类型

5.D.1 date

日期：年-月-日
日期 2008-08-08

5.D.2 time

时间：时：分：秒
时间 20:08:00

5.D.3 datetime

年-月-日 时：分：秒
2008-08-08 20:08:00

5.D.4 timestamp

存储的时间戳，就是 java 中 Date 类的 getTime()的值
1997-1-1 到当前的毫秒值

5.E 逻辑型

5.E.1 bit

只能取 0 和 1

5.F 注意：操作时字符型和日期需要单引号，数值型不需要

五 约束条件

都可建表时放到 数据类型后边

1.A 主键约束：

1.A.1 所约束的值必须唯一且非 NULL，不能重复

1.A.2 建表时： ***字段名 1 数据类型 primary key***

1.A.3 表已存在添加主键:

alter table table_name ***add primary key***(field);

1.A.4 自动增长字段: ***字段名 1 数据类型 primary key anto_increment***

只能主键或者有唯一索引的键才能设置 ***auto_increment，一个表只能由一个自增字段***

1.B 唯一约束(唯一索引？？)

1.B.1 ***字段名 1 数据类型 unique***

只能主键或者有唯一索引的键才能设置 ，可以为 NULL， ***auto_increment 一个表只能由一个自增字段***

1.C 非空约束

1.C.1 ***字段名 1 数据类型 not null***

1.D 外键约束

foreign KEY(ID) reference 表名(ID)

六　数据库操作

1　创建数据库

1.A　语法：create database [if not exists] 数据库名　[character set charset_name]

1.A.1　If not exists 可以防止报错，数据库不存在时才创建

1.A.2　create database db_book 创建 db 数据库

1.A.3　character set charset_name 设置编码

2　查看数据库

2.A　show databases

2.A.1　查看所有数据库，结尾 s

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| accountbook        |
| mydb1              |
| mysql              |
| performance_schema |
| test               |
+--------------------+
6 rows in set (0.00 sec)
```

2.B　show create database db_name

2.B.1　查看数据库的创建语句

```
mysql> show create database mydb1;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| mydb1    | CREATE DATABASE `mydb1` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.C 查看当前使用的数据库

2.C.1 select database(); 注意括号

3 修改数据库

3.A 数据库的名称无法修改，可修改数据库编码

3.B 格式： Alter database 数据库名 character set 编码名称;

3.C 例： Alter database mydb1 character set utf8;

4 删除数据库

4.A 格式：drop database [if exists] db_name;

5 选择数据库

5.A 格式：use db_name; 这里不加分号也行

七 表操作

1 创建表

1.A 语法：

```
create table table_name(
    字段名 1 数据类型,
    字段名 1 数据类型,
    …
    字段名 n 数据类型//最后一个字段末尾没有逗号

);//注意结尾有分号
```

1.B 例：

```
create table employee(
    id int,
    name varchar(10),
    gender varchar(1),
    birthday date,
    entry_date date,
    job varchar(30),
    salary double
    resume text
)
```

1.C 注意

最后一个字段末尾没有逗号

2 查看表

2.A 查看所有表

```
show tabes;
```

2.B 查看表结构

```
desc table_name;
```

2.C 查看建表语句

```
show create table table_name
```

3 修改表

3.A 增加列

3.A.1 alter table 表名 add *列名 数据类型;*

3.B 修改列(modify 修改数据类型)

3.B.1 alter table 表名 modify 列名 数据类型

```
alter table employee modify id int primary key auto_increment;
```
设置 id 字段为主键和字段增长

3.C 删除列

3.C.1 alter table 表名 drop 列名

3.D 修改列名

3.D.1 alter table 表名 change 旧列名 新列名 数据类型

3.E 修改表名

3.E.1 alter table 表名 rename to 新表名

3.E.2 reanme table 旧表名 to 新表名

3.F 修改列顺序

3.F.1 alter table 表名 modify 列名 1 数据类型 after 列名 2

将列名 1 放到列名 2 后 ,before 不行

3.G 修改表字符集

3.G.1 alter table 表名 character set 编码

关于mysql创建数据库时指定编码和创建表时指定编码的区别?

发布于 2018-01-22

▲ 赞同 2 ▼ 添加评论 分享 收藏 喜欢

Pader
PHP / FreeBSD / 操作系统和网络技术 / 高科技爱好者

1 人赞同了该回答

数据库的编码就是个默认选项而已，你创建表的时候如果不指定编码就会默认采用数据库的编码，如果你指定了，那你的表就是你指定的编码。两者之间没有必然的联系。就像字符串字段的编码你不指定的话默认采用表的编码一样。

不过你在表已经建好了之后，再去修改数据库的编码，表的编码是不会一起变的，因为表是建立的时候采用了数据库的编码值，就像填进去一样。

编辑于 2018-01-29

▲ 赞同 1 ▼ 添加评论 分享 收藏 喜欢

曾磊
多年软件开发，专注游戏开发5年+

泻药。

创建数据库时指定的编码，表示整个数据库用其作为默认编码。

创建表时，指定的编码，表示用该编码创建当前表（如果列没指定编码，就用这个）

当两者不一样时，以 创建 表时 指定的为准

简单讲就是，继承，覆盖的关系

发布于 2018-02-01

▲ 赞同 ▼ 添加评论 分享 收藏 喜欢

4 删除表

4.A.1 drop table 表名

八 记录操作

1 插入记录(insert)

1.A 语法：insert into 表名[(列名 1，列名 2，列名 3...)] values (value1，value2...);

1.B 练习：

向 employee2 表中添加 3 条记录

```
insert into employee2 (id,name,gender,birthday,entry_date,job,salary,resume)
values(null,'tom','m','1999-9-1','2020-10-10','doctor',5000,'good');

insert into employee2 values(null,'rose','w','2008-1-10','2019-8-10','teacher',3000,'good');

insert into employee2(id,name,gender) values(null,'jerry','m');

insert into employee2 values(null,'张飞','男','2008-1-10','2019-8-10','将军',3000,'猛男一个');
```

1.C 注意：

1.C.1 若值是字符串或日期，需要用单引号包裹

1.C.2 如果插入的字段为全部字段，列名可省略

1.C.3 若字段是自增，插入 null 或 0 值也没问题，也能自增

1.D 乱码问题

1.D.1 原因：如果安装时 mysql 选择的编码为 utf-8，而 cmd 窗口默认用的 GBK，即编码不一致导致。

1.D.2 解决 1(临时方法，当前窗口有效)：用 sql 语句，*set names gbk;*解决，只是告诉 mysql 当前数据为 gbk，不会改变 mysql 底层的编码存储方式。

1.D.3 解决 2(永久)：在 mysql 安装目录下有 my.ini 文件，修改 *default-character-set=gbk*，不会改变 mysql 底层的编码存储方式。

2 更新记录(update)

2.A 语法：update 表名 set 列名=值表达式[,列名=值表达式,...] [where 条件表达式]

2.B 练习：

```
将所有员工的薪水改为 2000
update employee2 set salary=2000;
将张飞的工资改为 1000
update employee2 set salary=1000 where name='张飞';
将张飞的工资在原来的基础上增加 500
update employee2 set salary=salary+500 where name='张飞';
```

3 删除记录(delete)

3.A 语法：delete from 表名 [where 条件表达式]

3.B 删除所有数据

3.B.1 delete from 表名; 一行行删除数据

3.B.2 turncate table 表名; 摧毁整个表重建建 数据太多的情况效率更高

4 查询记录(查询)

4.A 语法

```
SELECT <目标表的列名或表达式序列>
FROM <基本表名或视图序列>
[WHERE <行条件表达式>]
[GROUP BY <列名序列>]
        [HAVING <组条件表达式>]]
[ORDER BY <列名 [ASC|DESC]> ,...]
```

[ ]表示成分可有可无

4.B 函数

4.B.1 Ifnull(x,0):如果 x 为 null，返回数值 0，null 和任何数据相加为 null

4.C 聚合函数

4.C.1 注意

只对列聚合，不对行!
SELECT NAME,MAX(chinese) FROM exam;，这里写了 name，返回的名字不是最高分的人的名字，NAME 默认会返回所有行，但 max 只有一行。此时只能用嵌套查询

4.C.2 count()函数

求记录数，count 不会忽略 null，如果有 id 列有一个值为 null，那么 count(id)的记录时不会加上有 null 值的行.

4.C.3 avg/max/sum…等

计算 null 值时会忽略

4.C.4 练习：

统计一个班共多少学生

```
select count(*) from exam;
```

统计数学大于 75 分的学生有多少个

```
select count(*) from exam where math>75;
```

统计一个班的数学总成绩

```
select sum(math) from exam;
```

统计一个班的英语平均分

```
select avg(english) from exam;
select avg(ifnull(english,0)) from exam;
```

求班级总分的最高分和最低分

```
select max(chinese+math+ifnull(english,0)) from exam;
select min(chinese+math+ifnull(english,0)) from exam;
```

查询总分最多的学生

```
select * from exam where (chinese+math+ifnull(english,0))=(
select max(chinese+math+ifnull(english,0)) from exam);
```

4.D 别名

4.D.1 Select name [as] n from exam; mysql 中 as 可省略

4.E 去重 distinct

4.E.1 SELECT DISTINCT *column_name,column_name*

FROM *table_name*;

4.F Where 子句

4.F.1 下面的运算符可以在 WHERE 子句中使用：

运算符	描述
=	等于
<>	不等于。注释：在 SQL 的一些版本中， 该操作符可被写成 !=，mysql 中二者都可使用
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN	在某个范围内，mysql 包括二端的值

值 1 AND 值 2	
LIKE	搜索某种模式 like '66' 不使用通配符等同于 ='66',数组可以 like 66 如果有通配符，必须 加单引号 在 SQL 中，可使用以下通配符： % 替代 0 个或多个字符 _ 替代一个字符
IN	指定针对某个列的多个可能值

4.F.2 REGEXP/RLIKE

效果相同，前面可加 not

[*charlist*]	字符列中的任何单一字符
[^*charlist*] 或 [!*charlist*]	不在字符列中的任何单一字符

4.F.3 （1/true,0/false）不带比较运算符的 WHERE 子句：

WHERE 子句并不一定带比较运算符，当不带运算符时，会执行一个隐式转换。当 0 时转化为 false，1 转化为 true。例如：

```
SELECT studentNO FROM student WHERE 0
```

则会返回一个空集，因为每一行记录 WHERE 都返回 false。

```
SELECT studentNO FROM student WHERE 1
```

返回 student 表所有行中 studentNO 列的值。因为每一行记录 WHERE 都返回 true。

4.F.4 not

表达式结果取反，例如：not in... not between...
not math>85

4.F.5 exists

EXISTS 运算符用于判断查询子句是否有记录，如果有一条或多条记录存在返回 True，否则返回 False。

```
SELECT Websites.name, Websites.url
 FROM Websites
WHERE NOT EXISTS
  (SELECT count FROM access_log WHERE Websites.id = access_log.site_id
AND count > 200);
```

4.G ORDER BY 排序

4.G.1 注意：

Asc/Desc 只对当前字段有效
mysql 默认升序 Asc

4.G.2 order by math;

math 升序

4.G.3 order by math desc;

math 降序

4.G.4 order by math , engish desc;

先 math 升序，再 engish 降序

4.G.5 order by math desc , engish desc;

先 math 降序再 engish 降序

4.H GROUP BY 列名 HAVING

4.H.1 HAVING 是在分组后筛选，注意与 WHERE 区别，WHERE 是在分组前

九 数据库的备份与恢复

1.A 备份数据库

1.A.1 在 cmd 命令行，无需登录 mysql 管理员模式运行

Mysqldump –u<*用户名*> –p[*密码*] 数据库名称 >备份文件位置

账户 root 密码 admin

1.A.2 mysqldump -uroot -p mydb1>C:/1.sql 密码后输安全

Enter password: *****

1.A.3 mysqldump -uroot -padmin mydb1>C:/2.sql

1.B 还原数据库

1.B.1 在 cmd 命令行，无需登录 mysql 管理员模式运行

和备份差不多，但箭头是反方向

Mysqldump –u<*用户名*> –p[*密码*] 数据库名称 <备份文件位置

十　多表设计

1　一对一(1:1)

1.A　在任何一方保存加入另一方的主键作为外键

2　一对多(1:N)

2.A　在 N 那一方保存加入另一方的主键作为外键

3　多对多(M:N)

3.A　将他们联系的属性单独建表，然后把 M 和 N 表的主键作为新表的外键

十一　多表查询

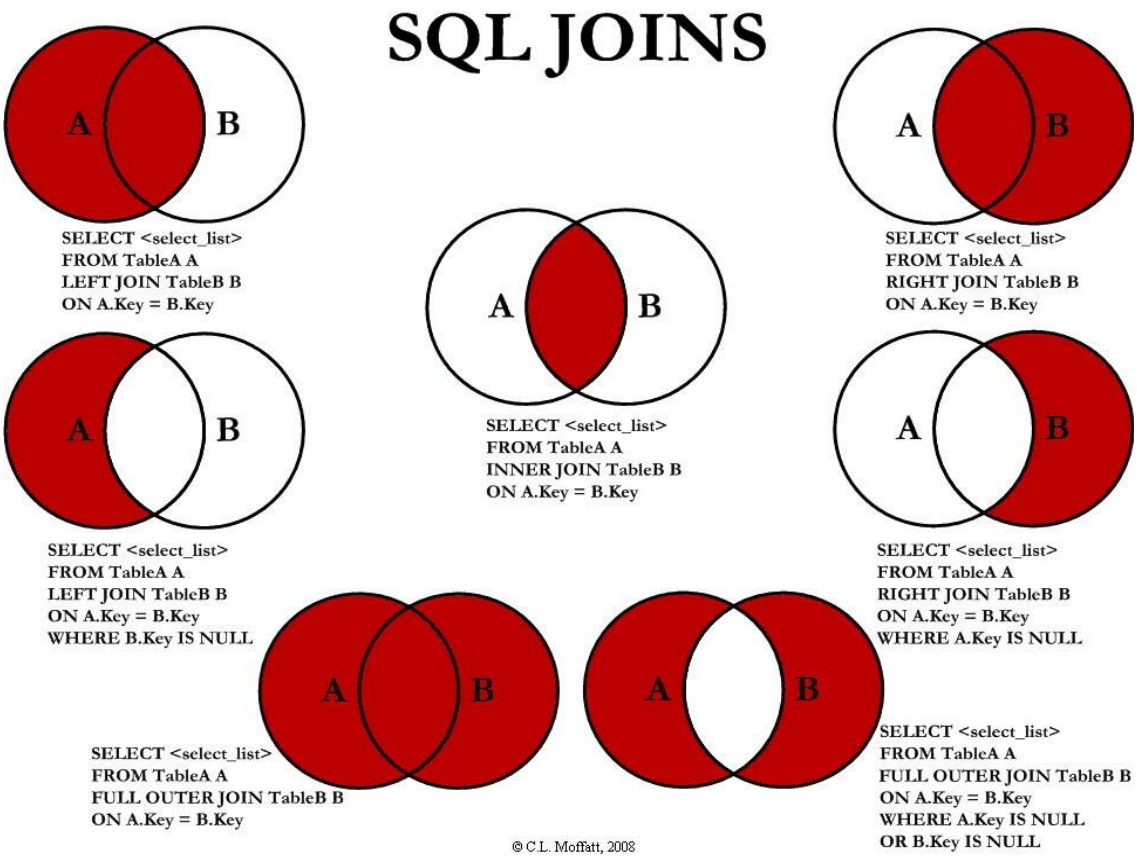
1　笛卡尔积(R x S)

2　连接(都是先做笛卡尔积在条件筛选？)

在使用 **join** 时，**on** 和 **where** 条件的区别如下：

- 1、 **on** 条件是在生成临时表时使用的条件，它不管 **on** 中的条件是否为真，都会返回左边表中的记录。
- 2、**where** 条件是在临时表生成好后，再对临时表进行过滤的条件。这时已经没有 **left join** 的含义（必须返回左边表的记录）了，条件不为真的就全部过滤掉。

MySQL 中不支持 FULL OUTER JOIN 提供了 union(需要相同数量的列名和相同的列名，是合并行)



2.A.1 内连接查询

Inner join …on …查询左边有，右边也有的数据，即交集
内连接 inner 可省略

2.A.2 左外连接查询

left join …on …在内连接基础上，加上左边表的数据
因为右表中没匹配，会以 null 显示

2.A.3 右外连接查询

right join …on …在内连接基础上，加上左边表的数据
因为左表中没匹配，会以 null 显示

2.A.4 全外连接

Full join …on…在内连接基础上，加上左边表和右边表数据，若不匹配以 null 显示
注意：mysql 不支持 full join on 但可以用 union

SQL UNION 语法(要求相同的列名，相同的列数)

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

注释：默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。

SQL UNION ALL 语法

```
SELECT column_name(s) FROM table1
UNION                                                    ALL
SELECT column_name(s) FROM table2;
```

注释：UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。

十二　数据库事务

1　事务的概念

事务指逻辑上的一组操作，要么全部成功，要么全部不成功
假如二条 sql 语句，要么一起执行成功，要么一起失败，

2　管理事务

数据库默认支持事务，数据库的事务是一条 SQL 语句被认为是一个事务

2.A　start transaction

开始一个事务

2.B　commit

以上事务一起提交执行
例如在 delete 删除某条记录时，commit 之前，只是增加删除标记，并未真正删除。

2.C　rollback

以上事务一起回退不执行

3　JDBC 中的事务控制

Connection 对象的三个方法
mysql 数据库默认执行 executeUpdate ()后，自动提交事务

3.A　conn.setAutoCommit(false)

自动提交为 false，相当于开启事务，等同于 sql 中的 start transaction

3.B　conn.commit()

提交事务

3.C conn.rollback

回退事务

3.D 实例：

```
Connection conn=null;
PreparedStatement pre=null;
ResultSet res=null;
try {

    conn= JDBCUtil.getConnection();
    //不要自动提交
  conn.setAutoCommit(false);

    pre = conn.prepareStatement("update users set account=? where
id=8");
    pre.setString(1,"DEF");
    pre.executeUpdate();

    int i = 1 / 0;

    pre = conn.prepareStatement("update users set account=? where
id=7");
    pre.setString(1,"abc");
    pre.executeUpdate();
    //如果设置了自动提交，而没有 commit，数据实际没变
  conn.commit();
} catch (SQLException e) {
    e.printStackTrace();
    if (conn!=null){
        try {
            //虽然 出现了异常 commit 没有执行，但是还是要 rollback 释放一下资源
            //同时结束当前事务
     conn.rollback();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

4 事务的四大特性(ACID)

4.A 原子性(Atomicity)

事务是一个不可分隔的工作单位，事务中的操作要么一起发送，要么一起不发生

4.B 一致性(Consistency)

事务前后数据的完整性必须保持一致

4.C 隔离性(lsolation)

多个事务同时访问数据库，一个事务不能被其它事务所干扰

4.D 持久性(Durability)

事务一旦提交，对数据库的改变是永久的、不可逆的。

5 隔离性(Isolation)

5.A 隔离性分析

多个用户同时查询同一条数据：无需隔离
多个用户同时修改同一条数据：需要隔离
一个用户查询，另一个用户修改数据:根据情况，讨论隔离级别

5.B 隔离性可能造成的问题

5.B.1 脏读

一个事务读到另一个事务未提交的数据，然后事务未提交 rollback 后，另一个事务读取的数据又是最开始的数据

设置隔离级别最低：set session transaction isolation level read uncommitted;
例 最开始 account=XX
事务一：
开启事务
Start transaction
修改 account =pp
update users set account='PP' where id=8;
事务二：
读取 数据 account 等于 PP

事务一：rollback

事务二：select 读取 account 等于 XX

5.B.2 不可重复读

一个事务读到另一个事务已提交的数据
和脏读顺序相反

5.B.3 虚读(幻读)

发生概率非常低，在读取整表时，多次读取的结果不一样

6 隔离级别

为防止以上问题，数据库提供了不同的隔离级别，越安全效率越低

6.A read uncommitted;

不做任何隔离

6.B read committed

可以防止脏读，但不能防止 不可重复读和虚读幻读

6.C repeatable read

可以防止脏读和不可重复读，但是不能防止虚度（幻读）

6.D serializable

可以防止所有隔离性问题,但是数据库被设置为了串行的数据库，性能低

一般用 B 和 C，mysql 默认 repeatable read
oracle 默认是 read committed 没有 repeatable read

7 操作数据库的隔离级别

7.A 查询

```
SELECT @@tx_isolation;
```

7.B 设置隔离级别

set <[session][global]> transaction isolation level 隔离级别;
默认是 session，表示当前连接有效,global 表示全局修改

8 数据库的锁(整个数据库)

8.A 共享锁

共享锁可以和共享锁共存，共享锁和排他锁不能共存，在 serializable 级别下查询时会添加共享锁，其他级别不会

8.B 排他锁

排它锁和任何锁不能共存，在任何隔离级别下做增删改操作都会添加排它锁

9 数据库死锁

彼此握着对方的资源互不释放，导致彼此都无法继续执行

10 数据库活锁

彼此互相谦让，导致彼此都无法继续执行

十三 项目中发现的问题点

1 需求：

我想把第一个图片里面的 categoryId 的值改为第二图中的 id 值，根据 tag 和 categroyName 的对应关系

Row: Rows in a Rang First Row: 0 No. of Rows: 50 Refresh

eDate	categoryId	user	visito	fileTyp	targetF	source	tag	desc
)4-03	1	0	4	.swf	/swf/20	/docx/	C	层叠
)4-03	1	0	9	.swf	/swf/20	/docx/	JS	JavaS
)4-07	1	0	8	.swf	/swf/20	/docx/	Jquery	AjQue
)4-08	1	0	122	.swf	/swf/20	/docx/	MySql	数据
)4-11	1	0	15	.swf	/swf/20	/docx/	JAVA	由于
)4-14	1	0	69	.swf	/swf/20	/docx/	TomCat	可以
)4-15	1	0	14	.swf	/swf/20	/docx/	Servlet	rvlet
)4-17	1	0	29	.swf	/swf/20	/docx/	JAVA	综合
)4-18	1	0	77	.swf	/swf/20	/docx/	会话	为了
)4-20	1	0	13	.swf	/swf/39	/docx/	AJAX	AJAX,
)4-21	1	0	4	.swf	/swf/6b	/docx/	MVC	MVC,M
)4-22	1	0	34	.swf	/swf/71	/docx/	Filter	Filte
)4-22	1	0	31	.swf	/swf/f6	/docx/	JAVA	监听
)4-22	1	0	40	.swf	/swf/98	/docx/	JAVA	设计
)4-22	1	0	23	.swf	/swf/8b	/docx/	MD5	加密
)4-23	1	0	94	.html	/html/2	/html/	JAVA	最近
)4-24	1	0	104	.html	/html/2	/html/	MySql	昨天
)4-27	1	0	16	.html	/html/2	/html/	JAVA	1、前
)4-27	1	0	12	.html	/html/2	/html/	Spring	一、
)4-27	1	0	10	.html	/html/2	/html/	Spring	一、
)4-28	1	0	7	.html	/html/2	/html/	Spring	一、
)4-28	1	0	4	.html	/html/2	/html/	Spring	一、
)4-28	1	0	9	.html	/html/2	/html/	JAVA	一、
)4-29	1	0	10	.html	/html/2	/html/	Spring	一、
)4-29	1	0	5	.html	/html/2	/html/	JAVA	〇、
	(NULL)	ULL)	0	(NULL)	(NULL)	(NULL)	(NULL)	(NULL

```
2   ALTER TABLE category CHANGE `type` categoryName VARCHAR(10) UNIQUE;
3
4   UPDATE articles SET categoryId=(SELECT * FROM categoryarticles
```

1 Messages | 2 Table Data | 3 Info | 4 History

All Row: Rows in a Rang First Row: 0 No. of Rows:

id	categoryName
1	JAVA
2	MM
3	AJAX
4	CSS
5	Filter
7	Jquery
8	JS
9	MD5
10	MVC
11	MySql
12	Servlet
13	Spring
14	TomCat
15	会话
(NULL)	(NULL)

2 实现

```
UPDATE articles a INNER JOIN category c ON  a.tag=c.categoryName SET categoryId=c.id;
```