## Sql的运算符优先级和一般的编程语言不一样



SELECT \* FROM `testscore` where `class` = 1 or `name` = '张四' and `score` = 92;

执行效果：



说明：sql执行先执行的`name` = '张四' and `score` = 92然后class` = 1（对所有记录来说，第一个条件都是false，第二个条件有三个记录是正确的）

## Sql事务中的四种隔离级别

### 读写分离

这些设置最基本的可以保障读写分离，至于更加负责的可以通过不同级别保障一件事物的处理逻辑正确。

### 对于一件事物的隔离需求

比如说我需要对外销售1张电影票，且登记一下销售信息到另一个表，至少需要以下3个步骤：

　　1.查询电影票数量是否满足销售1张电影票 SELECT remain\_count FROM cinema WHERE film\_id = 123456789;

　　2.更新电影票数量 UPDATE remain\_count = remain\_count -1 FROM cinema WHERE film\_id=123456789;

　　3.插入销售信息 INSERT INTO sell\_mes(id ,mes) values(id,mes);

　　试想一下如果我们其中的一步被出错了或者被其他操作打乱就很容易出现问题。比如说有两个销售系统A,B在销售同样的票，此时票只剩下1张，A接到订单要售出一张票，他查看电影票的数量大于1，于是要售出的时候，也就是在第一步执行完毕执行第二步的时候，B也接到订单，也看到余票大于1，B也要售出1张票。此时就出现了余票只有1张却售出两张的荒唐的情况出现。

　　所以一个良好的系统是需要通过ACID测试，至于ACID是什么，就自行百度吧，这篇博文讲的是隔离性的隔离级别以及例子

### 四种隔离级别

而事务的隔离级别有四种，隔离级别高的数据库的可靠性高，但并发量低，而隔离级别低的数据库可靠性低，但并发量高，系统开销小

#### 1.READ UNCIMMITTED（未提交读）

　　事务中的修改，即使没有提交，其他事务也可以看得到，比如说上面的两步这种现象就叫做脏读，这种隔离级别会引起很多问题，如无必要，不要随便使用

　　例子:还是售票系统，小明和小花是售票员，他们分别是两个不同窗口的员工，现在售票系统只剩下3张票，此时A来小华这里买3张票，B来小明买票，小华查到余票还有就给接了订单，就要执行第三步的时候，小明接到B的请求查询有没有余票。看到小华卖出了3张票，于是拒绝卖票。但是小华系统出了问题，第三步执行失败，数据库为保证原子性，数据进行了回滚，也就是说一张票都没卖出去。

　　总结:这就是事务还没提交，而别的事务可以看到他其中修改的数据的后果，也就是脏读。

#### 2.READ COMMITTED（提交读）

　　大多数数据库系统的默认隔离级别是READ COMMITTED，这种隔离级别就是一个事务的开始，只能看到已经完成的事务的结果，正在执行的，是无法被其他事务看到的。这种级别会出现读取旧数据的现象

　　例子:还是小明小华销售员，余票3张，A来小华那里请求3张订票单，小华接受订单，要卖出3张票，上面的销售步骤执行中的时候，B也来小明那里买票，由于小华的销售事务执行到一半，小明事务没有看到小华的事务执行，读到的票数是3，准备接受订单的时候，小华的销售事务完成了，此时小明的系统变成显示0张票，小明刚想按下鼠标点击接受订单的手又连忙缩了回去。

　　总结:这就是小华的事务执行到一半，而小明看不到他执行的操作，所以看到的是旧数据，也就是无效的数据

#### 3.REPEATABLE READ（可重复读）

　　REPEATABLE READ解决了脏读的问题，该级别保证了每行的记录的结果是一致的，也就是上面说的读了旧数据的问题，但是却无法解决另一个问题，幻行，顾名思义就是突然蹦出来的行数据。指的就是某个事务在读取某个范围的数据，但是另一个事务又向这个范围的数据去插入数据，导致多次读取的时候，数据的行数不一致。

　　例子:销售部门有规定，如果销售记录低于规定的值，要扣工资，此时经理在后端控制台查看了一下小明的销售记录，发现销售记录达不到规定的次数，心里暗喜，准备打印好销售清单，理直气壮和小明提出，没想到打印出来的时候发现销售清单里面销售数量增多了几条，刚刚好达到要求，气的经理撕了清单纸。原来是小明在就要打印的瞬间卖出了几张票，因此避过了减工资的血光之灾。

总结:虽然读取同一条数据可以保证一致性，但是却不能保证没有插入新的数据

#### 4.SERIALIZABLE（可串行化）

　　SERIALIZABLE是最高的隔离级别，它通过强制事务串行执行（注意是串行），避免了前面的幻读情况，由于他大量加上锁，导致大量的请求超时，因此性能会比较底下，再特别需要数据一致性且并发量不需要那么大的时候才可能考虑这个隔离级别

## MySQL时间格式转换

### 涉及的函数

[date\_format](https://wuzhuti.cn/tag/date_format/)(date, format) 函数，[MySQL日期格式化函数date\_format()](https://wuzhuti.cn/2367.html" \o "MySQL日期格式化函数date_format()" \t "_blank)

[unix\_timestamp](https://wuzhuti.cn/tag/unix_timestamp/)() 函数

[str\_to\_date](https://wuzhuti.cn/tag/str_to_date/)(str, format) 函数

[from\_unixtime](https://wuzhuti.cn/tag/from_unixtime/)(unix\_timestamp, format) 函数，[MySQL时间戳格式化函数from\_unixtime](https://wuzhuti.cn/2538.html" \o "MySQL时间戳格式化函数from_unixtime" \t "_blank)

### 时间转字符串

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select date\_format(now(), '%Y-%m-%d');**
3. #结果：2016-01-05

### 时间转时间戳

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select unix\_timestamp(now());**
3. #结果：1452001082

### 字符串转时间

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select str\_to\_date('2016-01-02', '%Y-%m-%d %H');**
3. #结果：2016-01-02 00:00:00

### 字符串转时间戳

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select unix\_timestamp('2016-01-02');**
3. #结果：1451664000

### 时间戳转时间

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select from\_unixtime(1451997924);**
3. #结果：2016-01-05 20:45:24

### 时间戳转字符串

**[sql]** [预览复制](https://wuzhuti.cn/2542.html)

1. **select from\_unixtime(1451997924,'%Y-%d');**
3. //结果：2016-01-05 20:45:24

### 附表

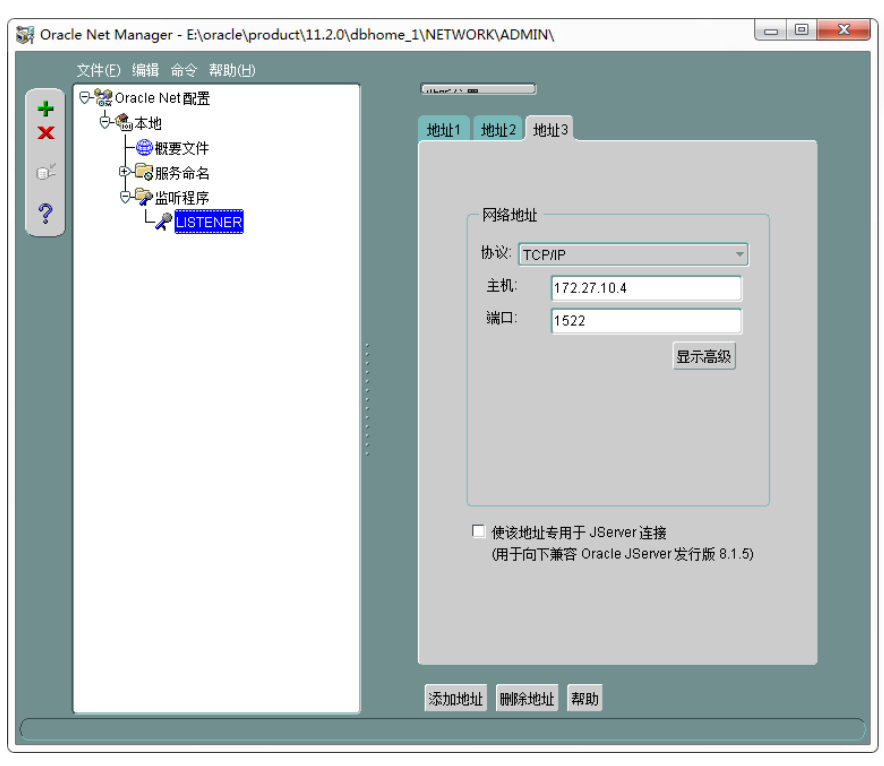
MySQL日期格式化（format）取值范围。

|  |  |  |
| --- | --- | --- |
|  | **值** | **含义** |
| 秒 | %S、%s | 两位数字形式的秒（ 00,01, ..., 59） |
| 分 | %I、%i | 两位数字形式的分（ 00,01, ..., 59） |
| 小时 | %H | 24小时制，两位数形式小时（00,01, ...,23） |
| %h | 12小时制，两位数形式小时（00,01, ...,12） |
| %k | 24小时制，数形式小时（0,1, ...,23） |
| %l | 12小时制，数形式小时（0,1, ...,12） |
| %T | 24小时制，时间形式（HH:mm:ss） |
| %r | 12小时制，时间形式（hh:mm:ss AM 或 PM） |
| %p | AM上午或PM下午 |
| 周 | %W | 一周中每一天的名称（Sunday,Monday, ...,Saturday） |
| %a | 一周中每一天名称的缩写（Sun,Mon, ...,Sat） |
| %w | 以数字形式标识周（0=Sunday,1=Monday, ...,6=Saturday） |
| %U | 数字表示周数，星期天为周中第一天 |
| %u | 数字表示周数，星期一为周中第一天 |
| 天 | %d | 两位数字表示月中天数（01,02, ...,31） |
| %e | 数字表示月中天数（1,2, ...,31） |
| %D | 英文后缀表示月中天数（1st,2nd,3rd ...） |
| %j | 以三位数字表示年中天数（001,002, ...,366） |
| 月 | %M | 英文月名（January,February, ...,December） |
| %b | 英文缩写月名（Jan,Feb, ...,Dec） |
| %m | 两位数字表示月份（01,02, ...,12） |
| %c | 数字表示月份（1,2, ...,12） |
| 年 | %Y | 四位数字表示的年份（2015,2016...） |
| %y | 两位数字表示的年份（15,16...） |
| 文字输出 | %文字 | 直接输出文字内容 |

## Oracle使用localhost连接可行，但使用ip不行，远程也不行问题

Oracle11g安装后只有本地可以连接，远程无法连接，而且本地只能配置成localhost配置成IP地址也无法连接。   
这是因为安装oracle的时候没有配置远程的监听，默认的监听是localhost。

1. 打开net manager
2. 配置监听端口



注意端口改成了1522了。