java+mysql 处理一天 24 小时时间问题

作者：centrexzj
原文链接：https://ld246.com/article/1536649971622
来源网站：链滴

问题描述

2个广告在一天的24个时间段播放，我们将一天分为24个小时，播放用1表示，不播放用0表示。组成2位的0和1组成的字符串。数据库存储为一个整数，通过位运算判断新加广告与原有广告是否有时间冲。

java代码在整数和24位0和1的字符串转化

```
import org.apache.commons.lang3.StringUtils;

public class NumberUtil {
    /**
     * int整数转换为4字节的byte数组
     *
     * @param i  整数
     * @return byte数组
     */
    public static byte[] intToByte4(int i) {
        byte[] targets = new byte[4];
        targets[3] = (byte) (i & 0xFF);
        targets[2] = (byte) (i >> 8 & 0xFF);
        targets[1] = (byte) (i >> 16 & 0xFF);
        targets[0] = (byte) (i >> 24 & 0xFF);
        return targets;
    }

    /**
     * long整数转换为8字节的byte数组
     *
     * @param lo  long整数
     * @return byte数组
     */
    public static byte[] longToByte8(long lo) {
        byte[] targets = new byte[8];
        for (int i = 0; i < 8; i++) {
            int offset = (targets.length - 1 - i) * 8;
            targets[i] = (byte) ((lo >>> offset) & 0xFF);
        }
        return targets;
    }

    /**
     * short整数转换为2字节的byte数组
     *
     * @param s    short整数
     * @return byte数组
     */
    public static byte[] unsignedShortToByte2(int s) {
        byte[] targets = new byte[2];
        targets[0] = (byte) (s >> 8 & 0xFF);
        targets[1] = (byte) (s & 0xFF);
```

原文链接：java+mysql 处理一天 24 小时时间问题

```
    return targets;
}

/**
 * byte数组转换为无符号short整数
 *
 * @param bytes   byte数组
 * @return short整数
 */
public static int byte2ToUnsignedShort(byte[] bytes) {
    return byte2ToUnsignedShort(bytes, 0);
}

/**
 * byte数组转换为无符号short整数
 *
 * @param bytes
 *            byte数组
 * @param off
 *            开始位置
 * @return short整数
 */
public static int byte2ToUnsignedShort(byte[] bytes, int off) {
    int high = bytes[off];
    int low = bytes[off + 1];
    return (high << 8 & 0xFF00) | (low & 0xFF);
}

/**
 * byte数组转换为int整数
 *
 * @param bytes
 *            byte数组
 * @param off
 *            开始位置
 * @return int整数
 */
public static int byte4ToInt(byte[] bytes, int off) {
    int b0 = bytes[off] & 0xFF;
    int b1 = bytes[off + 1] & 0xFF;
    int b2 = bytes[off + 2] & 0xFF;
    int b3 = bytes[off + 3] & 0xFF;
    return (b0 << 24) | (b1 << 16) | (b2 << 8) | b3;
}
public static byte[] getBooleanArray(byte b) {
    byte[] array = new byte[8];
    for (int i = 7; i >= 0; i--) {
        array[i] = (byte)(b & 1);
        b = (byte) (b >> 1);
    }
    return array;
}
public static String byteToBit(byte b) {
    return ""
```

```
                + (byte) ((b >> 7) & 0x1) + (byte) ((b >> 6) & 0x1)
                + (byte) ((b >> 5) & 0x1) + (byte) ((b >> 4) & 0x1)
                + (byte) ((b >> 3) & 0x1) + (byte) ((b >> 2) & 0x1)
                + (byte) ((b >> 1) & 0x1) + (byte) ((b >> 0) & 0x1);
    }

    /**
     * 将24位01字符串转化为整数
     * @param timeSeq 24位01字符串
     * @return 整数
     */
    public static  int str2Int(String timeSeq){
        timeSeq = "00000000"+timeSeq;
        int ret = -1;
        if(StringUtils.isEmpty(timeSeq)){
            return ret;
        }
        int length = timeSeq.length();
        if (length!=32){
            return ret;
        }
        byte[] intByteArray = new byte[4];
        for(int i=0;i<4;i++){
            String tmp = timeSeq.substring(i*8,i*8+8);
            intByteArray[i] =(byte)Integer.parseInt(tmp,2);
        }
        ret = byte4ToInt(intByteArray,0);
        return ret;
    }

    /**
     * 将整数转化为01字符串
     * @param timeCout
     * @return
     */
    public static String int2Str(int timeCout){
        byte[] c = intToByte4(timeCout);
        return byteToBit(c[1])+byteToBit(c[2])+byteToBit(c[3]);
    }
    public  static void main(String[] args){
        String a = "00000001000000000010000";
        System.out.println("Output: "+a);
        System.out.println("Output: "+str2Int(a));
        int  b = str2Int(a);
        System.out.println("Output: "+int2Str(b));
    }
}
```

mysql位运算

在mysql中，如果某条数据与其它数据存在一对多的关系，一般我们很自然的就会想到建立一个关系。例如有一个景点信息的数据表，其结构如下：

字段名	说明
id	int(主键)
name	varchar(景点名)
province	int(省份)
city	int(城市)

每个景点包含很多属性，例如适合旅游的月份，我们一般的做法可能有两种：一种是增加一个varcha字段，每个月份之间用一个特殊符号分隔保存，例如“1,2,3,11,12”；另一种方法是建立一个关系表如下：

字段名	描述
spots_id	int(景点ID)
month	int(适合月份，取值1-12)

第一种方法，查询极不方便，例如想查出适合2月份旅行的景点，就要使用like语句，效率极其低下。二种方法，只适合景点属性较少的场合。如果景点还包含其它属性，例如“高山”、“草原”之类的类属性，还有“美食”、“购物”等的主题属性，就要根据每个属性去建立一个关系表，扩展极其不，查询的时候可能需要联表查询，也影响效率。

我们知道，PHP当中的错误级别常量，是根据二进制位特性而确定的一个个整数，可以简单的通过位算定制PHP的错误报告。我们也可以将其应用到mysql当中，还是以上面的景点表为例，我们增加一字段，其结构如下：

字段名	描述
id	int(主键)
name	varchar(景点名)
province	int(省份)
city	int(城市)
month	int(适合旅行月份)

其建表语句为：

```
CREATE TABLE `spots` (

  `id` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT '景点ID',
  `name` varchar(50) NOT NULL COMMENT '景点名称',
  `province` int(5) unsigned NOT NULL COMMENT '景点所在省份',
  `city` int(11) unsigned NOT NULL COMMENT '景点所属城市',
  `month` int(11) unsigned NOT NULL DEFAULT '0' COMMENT '适合旅行的月份',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`),
  KEY `location` (`province`,`city`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='景点信息表'
```

注意：在这里不能使用1-12的数字来表示月份，而是使用1，2，4，8，16，32，64，128，512，104，2048，4096来表示。

以下为使用技巧：

- 当我们需要查询某个月份的景点时，例如查询3月份的景点，可使用以下语句：

```
SELECT * FROM `spots` WHERE `month` & 4 = 4
```

- 当设置某个景点适合某个月份时，例如设置4325的景点适合2月份，可使用下面的语句：

```
UPDATE `spots` SET `month` = `month` | 2 WHERE `id` = 4325
```

- 当取消设置某个景点的月份时，可使用下面的语句：

```
UPDATE `spots` SET `month` = `month` ^ 2 WHERE `id` = 4325
```

- 查询同时适合多个月份的数据，例如需要查询设置了11，12，1月份的景点，将其三个月份对应的值加起来，结果为6146，然后使用这个数值进行查询：

```
SELECT * FROM `spots` WHERE `month` & 6146 = 6146
```