# 1.mysql之TIMESTAMP（时间戳）用法详解

一、TIMESTAMP的变体

TIMESTAMP时间戳在创建的时候可以有多重不同的特性，如：

1.在创建新记录和修改现有记录的时候都对这个数据列刷新：

|  |  |
| --- | --- |
| 1 | TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP |

2.在创建新记录的时候把这个字段设置为当前时间，但以后修改时，不再刷新它：

|  |  |
| --- | --- |
| 1 | TIMESTAMP DEFAULT CURRENT\_TIMESTAMP |

3.在创建新记录的时候把这个字段设置为0，以后修改时刷新它：

|  |  |
| --- | --- |
| 1 | TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP |

4.在创建新记录的时候把这个字段设置为给定值，以后修改时刷新它：

|  |  |
| --- | --- |
| 1 | TIMESTAMP DEFAULT ‘yyyy-mm-dd hh:mm:ss' ON UPDATE CURRENT\_TIMESTAMP |

**MySQL目前不支持列的Default 为函数的形式,如达到你某列的默认值为当前更新日期与时间的功能,你可以使用TIMESTAMP列类型,**下面就详细说明TIMESTAMP列类型  
   
二、TIMESTAMP列类型

TIMESTAMP值可以从1970的某时的开始一直到2037年，精度为一秒，其值作为数字显示。  
TIMESTAMP值显示尺寸的格式如下表所示：  
：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | +---------------+----------------+  | 列类型　　　　| 显示格式　　　 |  | TIMESTAMP(14) | YYYYMMDDHHMMSS |  | TIMESTAMP(12) | YYMMDDHHMMSS　 |  | TIMESTAMP(10) | YYMMDDHHMM　　 |  | TIMESTAMP(8)　| YYYYMMDD　　　 |  | TIMESTAMP(6)　| YYMMDD　　　　 |  | TIMESTAMP(4)　| YYMM　　　　　 |  | TIMESTAMP(2)　| YY　　　　　　 |  +---------------+----------------+ |

“完整”TIMESTAMP格式是14位，但TIMESTAMP列也可以用更短的显示尺寸，创造最常见的显示尺寸是6、8、12、和14。  
你可以在创建表时指定一个任意的显示尺寸，但是定义列长为0或比14大均会被强制定义为列长14。  
列长在从1～13范围的奇数值尺寸均被强制为下一个更大的偶数。  
   
列如：

|  |  |
| --- | --- |
| 1  2  3  4  5 | 定义字段长度　　 强制字段长度  TIMESTAMP(0) ->　TIMESTAMP(14)  TIMESTAMP(15)->　TIMESTAMP(14)  TIMESTAMP(1) ->　TIMESTAMP(2)  TIMESTAMP(5) ->　TIMESTAMP(6) |

所有的TIMESTAMP列都有同样的存储大小，使用被指定的时期时间值的完整精度（14位）存储合法的值不考虑显示尺寸。不合法的日期，将会被强制为0存储

这有几个含意：

1.虽然你建表时定义了列TIMESTAMP(8)，但在你进行数据插入与更新时TIMESTAMP列实际上保存了14位的数据（包括年月日时分秒），只不过在你进行查询时MySQL返回给你的是8位的年月日数据。如果你使用ALTER TABLE拓宽一个狭窄的TIMESTAMP列，以前被“隐蔽”的信息将被显示。

2.同样，缩小一个TIMESTAMP列不会导致信息失去，除了感觉上值在显示时，较少的信息被显示出。

3.尽管TIMESTAMP值被存储为完整精度，直接操作存储值的唯一函数是UNIX\_TIMESTAMP()；由于MySQL返回TIMESTAMP列的列值是进过格式化后的检索的值，这意味着你可能不能使用某些函数来操作TIMESTAMP列（例如HOUR()或SECOND()），除非TIMESTAMP值的相关部分被包含在格式化的值中。  
例如，一个TIMESTAMP列只有被定义为TIMESTAMP(10)以上时，TIMESTAMP列的HH部分才会被显示，因此在更短的TIMESTAMP值上使用HOUR()会产生一个不可预知的结果。

4.不合法TIMESTAMP值被变换到适当类型的“零”值(00000000000000)。（DATETIME,DATE亦然）

例如你可以使用下列语句来验证：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | CREATE TABLE test ('id' INT (3) UNSIGNED AUTO\_INCREMENT, 'date1'  TIMESTAMP (8) PRIMARY KEY('id'));  INSERT INTO test SET id = 1;  SELECT \* FROM test;  +----+----------------+  | id | date1　　　　　|  +----+----------------+  |　1 | 20021114　　　 |  +----+----------------+  ALTER TABLE test CHANGE 'date1' 'date1' TIMESTAMP(14);  SELECT \* FROM test;  +----+----------------+  | id | date1　　　　　|  +----+----------------+  |　1 | 20021114093723 |  +----+----------------+ |

你可以使用TIMESTAMP列类型自动地用当前的日期和时间标记INSERT或UPDATE的操作。  
如果你有多个TIMESTAMP列，只有第一个自动更新。自动更新第一个TIMESTAMP列在下列任何条件下发生：  
  
1.列值没有明确地在一个INSERT或LOAD DATA INFILE语句中指定。  
2.列值没有明确地在一个UPDATE语句中指定且另外一些的列改变值。（注意一个UPDATE设置一个列为它已经有的值，这将不引起TIMESTAMP列被更新，因为如果你设置一个列为它当前的值，MySQL为了效率而忽略更改。）  
3.你明确地设定TIMESTAMP列为NULL.  
4.除第一个以外的TIMESTAMP列也可以设置到当前的日期和时间，只要将列设为NULL，或NOW()。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | CREATE TABLE test (  'id' INT (3) UNSIGNED AUTO\_INCREMENT,  'date1' TIMESTAMP (14),  'date2' TIMESTAMP (14),  PRIMARY KEY('id')  );  INSERT INTO test (id, date1, date2) VALUES (1, NULL, NULL);  INSERT INTO test SET id= 2;  +----+----------------+----------------+  | id | date1　　　　　| date2　　　　　|  +----+----------------+----------------+  |　1 | 20021114093723 | 20021114093723 |  |　2 | 20021114093724 | 00000000000000 |  +----+----------------+----------------+ |

第一条指令因设date1、date2为NULL,所以date1、date2值均为当前时间第二条指令因没有设date1、date2列值，第一个TIMESTAMP列date1为更新为当前时间，而二个TIMESTAMP列date2因日期不合法而变为“00000000000000”

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | UPDATE test SET id= 3 WHERE id=1;  +----+----------------+----------------+  | id | date1　　　　　| date2　　　　　|  +----+----------------+----------------+  |　3 | 20021114094009 | 20021114093723 |  |　2 | 20021114093724 | 00000000000000 |  +----+----------------+----------------+ |

这条指令没有明确地设定date2的列值，所以第一个TIMESTAMP列date1将被更新为当前时间

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | UPDATE test SET id= 1,date1=date1,date2=NOW() WHERE id=3;  +----+----------------+----------------+  | id | date1　　　　　| date2　　　　　|  +----+----------------+----------------+  |　1 | 20021114094009 | 20021114094320 |  |　2 | 20021114093724 | 00000000000000 |  +----+----------------+----------------+ |

这条指令因设定date1=date1，所以在更新数据时date1列值并不会发生改变而因设定date2=NOW()，所以在更新数据时date2列值会被更新为当前时间此指令等效为：

|  |  |
| --- | --- |
| 1 | UPDATE test SET id= 1,date1=date1,date2=NULL WHERE id=3; |

因MySQL返回的 TIMESTAMP 列为数字显示形式，你可以用DATE\_FROMAT()函数来格式化 TIMESTAMP 列，如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | SELECT id,DATE\_FORMAT(date1,'%Y-%m-%d %H:%i:%s') As date1,  DATE\_FORMAT(date2,'%Y-%m-%d %H:%i:%s') As date2 FROM test;  +----+---------------------+---------------------+  | id | date1　　　　　　　 | date2　　　　　　　 |  +----+---------------------+---------------------+  |　1 | 2002-11-14 09:40:09 | 2002-11-14 09:43:20 |  |　2 | 2002-11-14 09:37:24 | 0000-00-00 00:00:00 |  +----+---------------------+---------------------+  SELECT id,DATE\_FORMAT(date1,'%Y-%m-%d') As date1,  DATE\_FORMAT(date2,'%Y-%m-%d') As date2 FROM test;  +----+-------------+-------------+  | id | date1　　　 | date2　　　 |  +----+-------------+-------------+  |　1 | 2002-11-14　| 2002-11-14　|  |　2 | 2002-11-14　| 0000-00-00　|  +----+-------------+-------------+ |

在某种程度上，你可以把一种日期类型的值赋给一个不同的日期类型的对象。  
然而，而尤其注意的是：值有可能发生一些改变或信息的损失：  
   
1.如果你将一个DATE值赋给一个DATETIME或TIMESTAMP对象，结果值的时间部分被设置为'00:00:00'，因为DATE值中不包含有时间信息。　　  
2.如果你将一个DATETIME或TIMESTAMP值赋给一个DATE对象，结果值的时间部分被删除，因为DATE类型不存储时间信息。  
3.尽管DATETIME, DATE和TIMESTAMP值全都可以用同样的格式集来指定，但所有类型不都有同样的值范围。

例如，**TIMESTAMP值不能比1970早，也不能比2037晚，这意味着，一个日期例如'1968-01-01'，当作为一个DATETIME或DATE值时它是合法的，但它不是一个正确TIMESTAMP值！并且如果将这样的一个对象赋值给TIMESTAMP列，它将被变换为0**。     
   
三、当指定日期值时，当心某些缺陷：

1.**允许作为字符串指定值的宽松格式能被欺骗。**例如，因为“:”分隔符的使用，值'10:11:12'可能看起来像时间值，但是如果在一个日期中使用，上下文将作为年份被解释成'2010-11-12'。值'10:45:15'将被变换到'0000-00-00'，因为'45'不是一个合法的月份。  
   
2.**以2位数字指定的年值是模糊的，因为世纪是未知的。**MySQL使用下列规则解释2位年值：  
在00-69范围的年值被变换到2000-2069。 在范围70-99的年值被变换到1970-1999。

**PS：这里再为大家推荐一个本站Unix时间戳转换工具，附带了各种语言下时间戳的操作方法，包括PHP、MySQL、SQL Server、java等的时间戳获取与转换操作技巧：**