Atitit 数据迁移 表结构的问题

视图

函数迁移

目录

[1.1. 有两种方法可以解决此问题：SET GLOBAL log\_bin\_trust\_function\_creators = 1; 1](#_Toc16769)

[1.2. 导入sql模式 2](#_Toc3195)

[1.3. Navicate复制也可 3](#_Toc14952)

## 有两种方法可以解决此问题：SET GLOBAL log\_bin\_trust\_function\_creators = 1;

在MySQL控制台中执行以下命令：

SET GLOBAL log\_bin\_trust\_function\_creators = 1;

将以下内容添加到mysql.ini配置文件中：

log\_bin\_trust\_function\_creators = 1;

该设置放松了对不确定功能的检查。非确定性函数是修改数据（即具有更新，插入或删除语句）的函数。有关更多信息，请参见[此处](http://dev.mysql.com/doc/refman/5.0/en/stored-programs-logging.html)。

请注意，如果未启用二进制日志记录，则此设置不适用。

[存储程序的二进制记录](http://dev.mysql.com/doc/refman/5.0/en/stored-programs-logging.html)

如果未启用二进制日志记录，则log\_bin\_trust\_function\_creators不适用。

[log\_bin\_trust\_function\_creators](http://dev.mysql.com/doc/refman/5.0/en/server-system-variables.html" \l "sysvar_log_bin_trust_function_creators)

启用二进制日志记录时，此变量适用。

最好的方法是更好地理解和使用存储函数的确定性声明。MySQL使用这些声明来优化复制，因此，仔细选择它们以确保复制正常是一件好事。

****确定性**** 如果例程对于相同的输入参数总是产生相同的结果，则例程被认为是“确定性”，否则，则不是“确定性”。这主要用于字符串或数学处理，但不仅限于此。

****非确定性**** 与“确定性”相反。“ *如果例程定义中未给出DETERMINISTIC或NOT DETERMINISTIC，则默认值为NOT DETERMINISTIC。要声明函数是确定性的，必须显式指定DETERMINISTIC。* ” 因此，似乎如果没有声明，MySQl会将函数视为“ NOT DETERMINISTIC”。手册中的这一陈述与手册另一领域的其他陈述相抵触：*创建存储函数时，必须声明它是确定性的或未修改数据。否则，对于数据恢复或复制可能是不安全的。默认情况下，要接受CREATE FUNCTION语句，必须显式指定DETERMINISTIC，NO SQL或READS SQL DATA中的至少一个。否则会发生错误* “

如果没有声明，我个人会在MySQL 5.5中出错，因此无论我可能有其他声明如何，我总是至少放置一个声明“ DETERMINISTIC”，“ NOT DETERMINISTIC”，“ NO SQL”或“ READS SQL DATA”。

****READS SQL DATA**** 这明确告诉MySQL该函数将仅从数据库中读取数据，因此，它不包含修改数据的指令，但包含读取数据的SQL指令（即SELECT）。

****MODIFIES SQL DATA**** 这表明例程包含可能写入数据的语句（例如，它包含UPDATE，INSERT，DELETE或ALTER指令）。

****NO SQL**** 这表示例程不包含SQL语句。

**CONTAINS SQL** 这表示该例程包含SQL指令，但不包含读取或写入数据的语句。如果未明确给出这些特征，则为默认设置。此类语句的示例为SELECT NOW（），SELECT 10 + @ b，SET @x = 1或DO RELEASE\_LOCK（'abc'），它们执行但既不读取也不写入数据。

请注意，有些MySQL函数不是确定性安全的，例如：NOW（），UUID（）等，它们可能在不同的机器上产生不同的结果，因此包含此类指令的用户函数必须声明为NOT DETERMINISTIC 。同样，从未复制模式中读取数据的函数显然是NONDETERMINISTIC。\*

*例程性质的评估基于创建者的“诚实”：MySQL不检查声明为DETERMINISTIC的例程是否没有产生不确定结果的语句。但是，错误声明例程可能会影响结果或影响性能。将不确定的例程声明为DETERMINISTIC可能会导致优化器做出错误的执行计划选择，从而导致意外结果。将确定性例程声明为NONDETERMINISTIC可能会导致不使用可用的优化，从而降低性能。*

## 导入sql模式

遇到错误继续，，去掉事务

## Navicate复制也可