工具开发目的：便于数据库方面的日常开发，减少重复工作量

工具主要功能：1，升级检查脚本生成

2，升级脚本生成

3，存储过程参数字符串的格式化转换

详细功能说明如下：

一，《升级检查脚本生成》功能使用说明

工具根据update脚本生成检查脚本，主要目的是检查本次升级增加的表、表中增加的列、增加的存储过程等是否存在，升级后版本号是否正确。因此update脚本书写时，需要遵循如下格式规则：

1. 新建表相关：
2. 所有建表语句，表的名字要与关键字create table 在同一行
3. Create table 语句的左圆括号’(’要另起一行
4. 每一列语句的定义独自占用一行
5. 每一列语句的定义都需要定义为not null，并且定义默认值
6. primary key 语句独自占用一行
7. 建表结束符的分号需要与右圆括号’)’同在一行

例：

create table if not exists t\_char

(

aid bigint unsigned not null auto\_increment,

accname varchar(80) binary not null default '',

charname varchar(50) binary not null default '', charguid bigint unsigned not null default 0xffffffffffffffff,

primary key(aid)

)ENGINE = INNODB;

1. 表修改相关
2. alter table 语句 需要与表的名字在同一行
3. 每一列的增加、修改、删除的语句占用独立的一行，并且新加的列都需要定义为not null，也需要有默认值
4. 增加列时使用add column语句，删除列使用drop column语句，修改列定义使用modify column语句

例：

alter table t\_char

add column level smallint not null default 0,

drop column isvalid,

modify column level smallint unsigned not null default 0;

1. 其他

所有复合关键字之间使用一个空格来间隔

例：

create table，not null，bigint unsigned ……

二，《升级脚本生成》功能使用说明

工具根据比较新旧两个建库脚本来生成升级脚本，因此对于建库脚本需要遵循如下格式规则：

1. 新建表相关
2. 所有新建表之前需要添加一条drop table 语句
3. 表的名字要与关键字create table 在同一行
4. Create table 语句的左右圆括号需要各自另起一行
5. 表中的每一列的定义独自占用一行
6. 每一列语句的定义都需要定义为not null，并且定义默认值
7. primary key 语句的定义独自占用一行，无论主键中包含多少列
8. 如果在表中直接定义索引，那么每个索引的定义独自占用一行，无论索引中包含多少列，并且索引需要定义名字
9. 建表结束符的分号需要与右圆括号’)’同在一行
10. 为了便于解析，列名字不能包含use、index等关键字
11. 表中只有aid是自增的，不允许其他列定义 auto\_increment
12. 所有复合关键字之间使用一个空格来间隔
13. 如果表中有自增的列，那么列名需要为aid

例：

drop table if exists t\_gmuser;

create table t\_gmuser

(

aid bigint unsigned not null auto\_increment,

account varchar(50) not null default '',

pwd varchar(64) not null default '',

auth varchar(64) not null default '',

primary key(aid),

unique index index\_gmuser\_account(account)

);

2，创建索引相关

1. create index 关键字需要与索引名字、表的名字都在同一行
2. 索引的左右圆括号都各自另起一行
3. 索引中每一列，独自占用一行
4. 结束符分号与右圆括号同在一行
5. 所有复合关键字之间使用一个空格来间隔

例：

create unique index Index\_char\_charame on t\_char

(

charname

);

create index Index\_char\_level on t\_char

(

Level,

isvalid

);

1. 创建存储过程相关
2. 每个存储过程编写之前均以delimiter // 开始，以 delimiter ; 结束
3. 每个存储过程创建之前都要先drop 一下
4. 每个create procedure 语句不需要if not exists关键字，后面直接跟存储过程名字即可
5. 如果存储过程有参数，每个参数的定义独自占用一行
6. 如果有参数，参数列表与函数体的分隔符左右括号，各自独占一行
7. 函数体以独自占用一行的begin开始，以独自占用一行的 end// 或 end;// 结束
8. 所有复合关键字之间使用一个空格来间隔

例：

delimiter //

drop procedure if exists p\_gmuser\_add\_proc//

create procedure p\_gmuser\_add\_proc

(

paccount varchar(50),

ppwd varchar(64),

pauth varchar(64)

)

begin

declare vret int default 0;

declare vacc varchar(50) default '';

start transaction;

select account into vacc from t\_gmuser where account = paccount;

if vacc = paccount then

set vret = 2;

else

set vret = 1;

insert into t\_gmuser(account, pwd, auth) values(paccount, ppwd, pauth);

end if;

commit;

select vret;

end//

delimiter ;

1. 创建函数相关
2. 每个函数编写之前均以delimiter // 开始，以 delimiter ; 结束
3. 每个函数创建之前都要先drop 一下
4. 每个create function语句不需要if not exists关键字，后面直接跟函数名字即可
5. 如果函数有参数，每个参数的定义独自占用一行
6. 如果有参数，参数列表与函数体的分隔符左右括号，各自独占一行
7. 返回值类型的定义以 returns 关键字来定义
8. 函数体以独自占用一行的begin开始，以独自占用一行的 end// 或 end;// 结束
9. 所有复合关键字之间使用一个空格来间隔

例：

delimiter //

drop function if exists gm\_hexstr2uint\_f;//

create function gm\_hexstr2uint\_f

(

h varchar(8)

)

returns int unsigned

DETERMINISTIC

begin

declare result int unsigned default 0;

declare len int default 0;

declare i int default 1;

declare base bigint default 1;

declare tmp char(2);

set len = length(h);

while i <= len do

set tmp = substring(h, i, 2);

set result = result + conv(tmp, 16, 10) \* base;

set i = i + 2;

set base = base \* 256;

end while;

return result;

end;//

delimiter ;

1. 索引相关
2. 同一个库里，所有表的索引不允许重名
3. 索引的命名最后规则为 index\_表名\_列名
4. 注释相关
5. 注释不要与代码在同一行
6. 以#开头或是以/\*开头都行都被作为注释忽略