Mysql 业务数据预埋及清理

# 一、数据预埋

数据预埋的要求：根据业务逻辑生成合理的数据记录批量导入数据库中。

通过调研，初步拟定以下几种方案（**先讨论方法，效率问题待定**）：

## 1.1 Insert 一次插入多条数据

**语法：**

|  |
| --- |
| ***mysql> INSERT INTO tbl\_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9),(10,11,12)...;*** |

在程序中可以通过循环，添加Values对应的列表，最后使用一次executeUpdate完成插入操作。  
**缺点：**Mysql语句并不是越长越好，**MYsql语句长度有限制**，可以查看mysql的配置文件my.in中max\_allowed\_packet属性，并进行相应设置。

## 1.2 Load 从文本文档导入数据

首先通过程序生成包含业务逻辑数据的txt文本文档，注意编码应该为UTF-8格式。

**语法：**

|  |
| --- |
| *mysql>* ***Load Data InFile 'D:/1.txt' Into Table `res\_type\_collect` fields terminated by ',' lines terminated by '\n'*** |

其中'D:/1.txt'是你文件存放的绝对路径，注意是“/”而不是“\”；Table 后的值是你的表名；fields terminated by后的值是列与列间的分割标识（个人感觉“,”比较靠谱，不易出错。而tab可能在导入的时候出现问题）；lines terminated by后面的值是行与行之间的分割标识，一般都是“\n”.

**缺点：**

## 1.3存储过程自动生成数据

**语法：**

|  |
| --- |
| ***CREATE PROCEDURE generateData()***  ***BEGIN***  ***DECLARE v1 INT DEFAULT 1;***  ***WHILE v1 < 200 DO***  ***insert into customers (cust\_id,cust\_name) values(v1+1,'tester');***  ***set v1=v1+1;***  ***END WHILE;***  ***END***  **COMMIT;**  **CALL generateData;//执行事务** |

**缺点：**事务需要控制大小，事务太大也可能会影响执行效率，MySQL有innodb\_buffer\_size配置项，超过这个值会把innodb的数据刷到磁盘中，这时，效率会有所下降。

注意：MySQL默认的存储引擎是MyISAM，不支持事务处理；InnoDB存储引擎支持事务处理。InnoDB表关闭自动事务提交可以大大提高数据插入的效率。

## 1.4利用powerdesigner工具逆向生成预埋数据

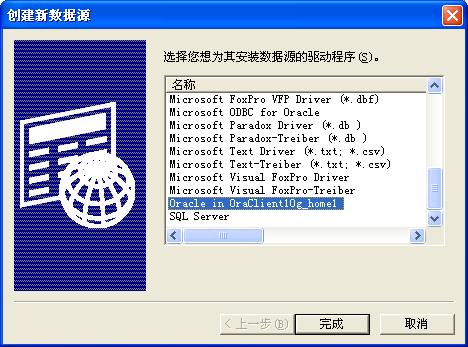
**摘自：POWERDESIGNER生成预埋数据 使用说明书**

步骤：

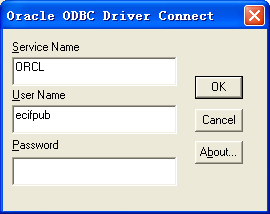
1. **配置一个ODBC数据源**

进入控制面板—管理工具，双击数据源（ODBC）快捷方式，进入ODBC配置界面

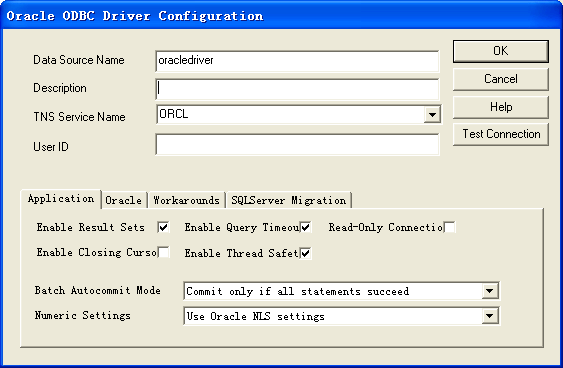
点击添加按钮，进入创建数据源界面，选中**Oracle10g**数据库的驱动程序，（前提条件是安装**Oracle10g**数据库的客户端），点击完成。（**编者注：**mysql用户选择mysql数据库的驱动程序即可）



进入**Oracle ODBC Drive**r配置界面，输入数据源名称、输入TNS服务名即SID，输入UserID，点击TestConnection，进入下面界面，输入密码，点击OK



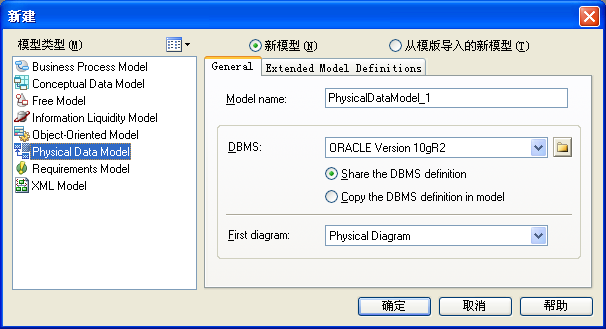
点击OK



至此ODBC数据源配置完毕。

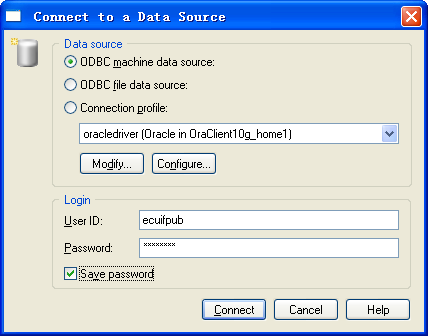
**2．通过逆向工程将关系数据库，生成物理数据模型（PDM）**

打开PowerDesigner软件，点击文件—新建按钮，进入新建模型界面



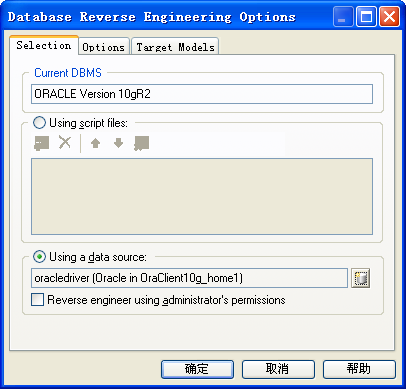
选择模型类型为Physical data Model ，选择DBMS为Oracle 10g，点击确定按钮。

从菜单栏选择数据库---连接数据源选项，进入连接数据源界面

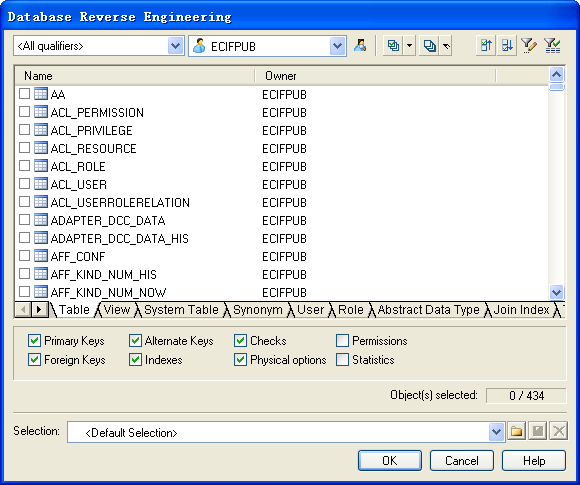


在Data source 中选择前面配置的驱动程序，这里选择的是**oracledriver**，输入用户ID和密码，点击连接。

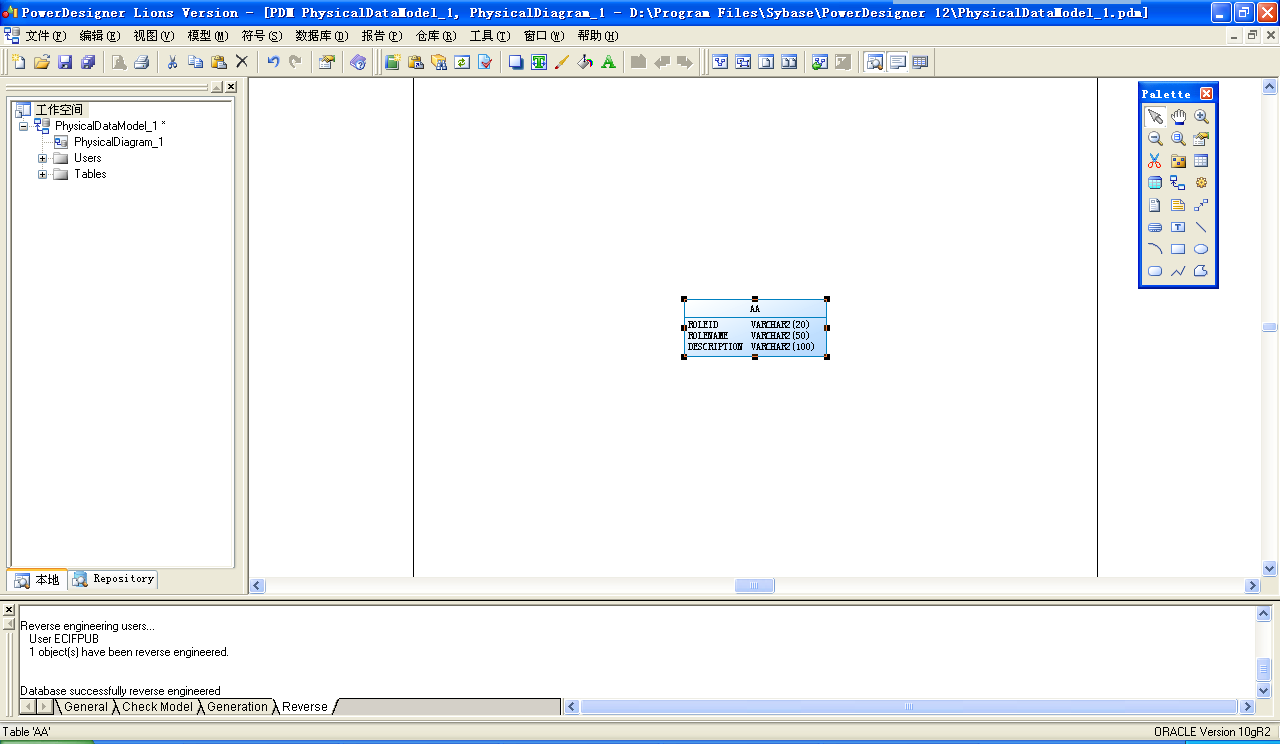
从菜单栏中选择数据库---逆向工程数据库（reverse engineer database），进入下面的界面



选择一个数据源，这里选择前面配置好的**oracledriver**，点击确定按钮，进入下面界面，此界面显示此数据库所有表、视图、用户，根据自己需要选择，这里以数据表为例

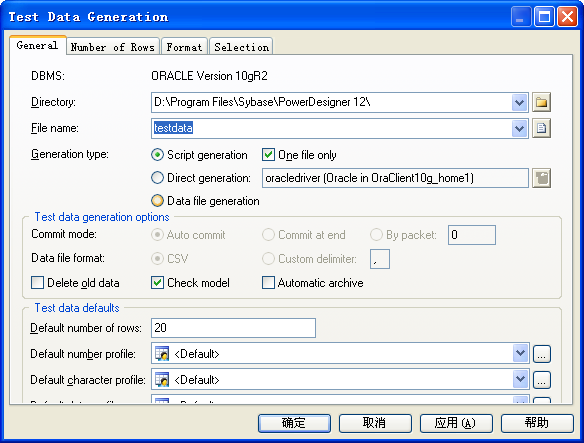


选择自己需要的数据库表，点击OK，这样PDM就生成了。如图所示

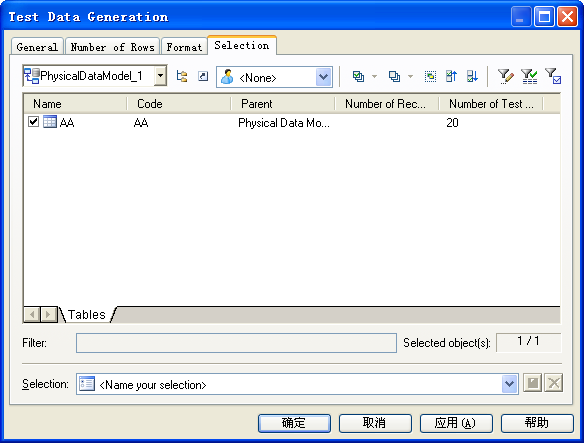


1. **从PDM生成测试数据**

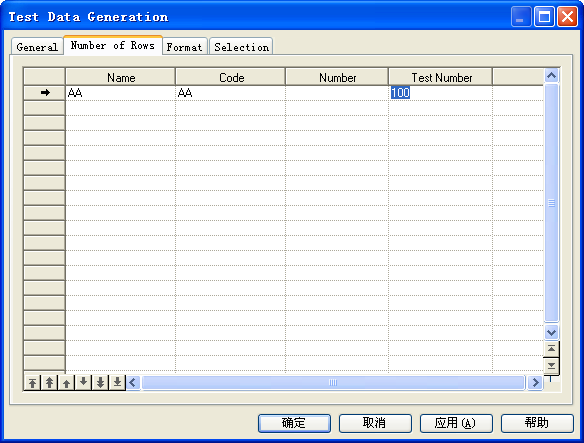
选择菜单栏—数据库—生成测试数据，进入下面的界面，输入文件名，选择存放路径，选择” Generation type”为” Script generation”，则生成SQL脚本；选择” Generation type”为” Direct generation”，则直接插入数据库；选择” Generation type”为” Data file generation”，则生成数据文件。建议通过生成SQL脚本的方式来预埋数据，因为如果预埋大量的数据，需要的时间可能会很长，而一旦发生错误，通过SQL脚本再执行一遍就可以了，下面就以此为例进行演示：



选择Selection页签，选择要生成测试数据的表，这里选择一个AA，点击确定按钮。



选择行数（Number of Rows）页签，进入下面界面，输入生成测试数据的行数，这里TestNumber取100作为例子，点击确定。



至此测试数据的sql脚本就生成了。

**缺点：**窗体程序，不知道是否支持纯命令行操作。

## 1.5比较分析

提高数据插入效率的基本原则如下：

1. 批量插入数据的效率比单数据行插入的效率高；
2. 插入无索引数据表比插入有索引数据表快一些；
3. 较短的SQL语句的数据插入比较长的语句快；

### insert vs load

[www.itput.net/thread-510339-1-1.html](http://www.itput.net/thread-510339-1-1.html) 做了insert 和load 批量 插入数据的性能测试

结论如下：

1. 随着数据量的成倍增加，load的性能与insert得性能越来越接近，不过仍然**优于insert**。
2. 采用分文件的形式可以有效提高load方法的执行效率，但是对insert方法没有什么提高

究其根源是因为MySQL内部对于load 和insert的处理机制不同。Load的处理机制是，在执行load之前，会关掉索引，当load全部执行完毕之后，再重新创建索引；insert处理机制是，每插入一条则更新一次数据库，更新一次索引。

另外，还体现在load省去了sql语句解析，sql引擎处理，而是直接生成文件数据块，所以会比insert快很多

Insert改进：

|  |
| --- |
| LOCK TABLES ‘TABLENAME’ WIRTE;  INSERT INTO ‘TABLENAME’ VALUES (‘1’,’2’),(‘3’,’4’);  INSERT INTO ‘TABLENAME’ VALUES (‘1’,’2’),(‘3’,’4’);  INSERT INTO ‘TABLENAME’ VALUES (‘1’,’2’),(‘3’,’4’);  UNLOCK TABLES ‘TABLENAME’; |

使用该方法可以获得与load方法差不多的效率，因为直到所有的insert语句都完成之后，索引缓存一次性刷新到磁盘中。且**insert数据可靠，load data会产生格式问题，甚至记录末尾插入位置字符**。（一家之言？）

Insert 一次插入多条数据比多条insert语句更快，因为减少了语句总数，最大程度上减少了索引刷新的次数。

可以通过增加系统变量key\_buffer\_size的值使得load和insert更快。

[www.3lian.com/edu/2013/07-15/80916.html](http://www.3lian.com/edu/2013/07-15/80916.html)

实验比较了简单的insert into、优化的SQLinsert(即一次插入多条数据)、事务提交三种方法。

结论如下：

执行速度上，优化insert(1min)>事务(8min)>简单的insert into(2h)

# 二、数据清理

## 2.1清空数据表

Mysql 数据表的删除清理主要有truncate、delete和drop三种关键字

不同点：

1. Truncate 和 delete只删除数据不删除表的结构（定义）；drop语句将删除表的结构被依赖的约束（constrain）、触发器（trigger）、索引（index）；依赖于该表的存储过程/函数将保留，但是变成invalid状态。
2. Delete语句是数据库操作语言（DML），这个操作会放到rollback segment中，事务提交之后才生效；如果有相应的trigger，执行的时候将被触发。**Truncate、drop**是数据库定义语言（DDL），操作立即生效，原数据不放到rollback segment中，**不能回滚**，操作不触发trigger
3. Delete语句不影响表所占有的extent，高水线（high watermark）保持原位置不懂；drop语句将表所占用的空间全部释放；truncate语句缺省情况下将空间释放到minextents，除非使用reuse storage；truncate会将高水线复位（回到最开始）。
4. **速度上，一般来讲，drop > truncate > delete**
5. 安全性：小心使用drop和truncate，尤其没有备份的时候，否则哭都来不及。使用上，想删除**部分**数据行用delete，注意带上where子句。回滚段要足够大。想删除表，用drop。想保留表而将所有数据删除，如果和事务无关，用truncate即可，如果和事务有关或者想触发trigger，还是用delete。如果是整理表内部的碎片，可以用truncate跟上reuse storage，再重新导入/插入数据
6. 对于有foreign key约束引用的表，不能使用truncate ，而应该使用不带where子句的delete。有外键约束时，delete [Users] DBCC CHECKIDENT(‘Users’,RESEED,0);其中Users是表名。
7. Truncate不能用于参加了索引视图的表

## 2.2清空数据库

方法1：重建库和表

只导出表结构

导出整个数据库结构（不包含数据）

Mysqldump –h localhost –uroot –p123456 **–d** database >dump.sql

导出单个数据表结构（不包含数据）

Mysqldump –h localhost –uroot –p123456 **–d** database **tablename**>dump.sql

只导出表数据

导出整个数据库数据

Mysqldump –h localhost –uroot –p123456 **–t** database >dump.sql

导出结构+数据

导出整个数据库结构和数据

Mysqldump –h localhost –uroot –p123456 database >dump.sql

导出单个数据表结构和数据

Mysqldump –h localhost –uroot –p123456 database tablename>dump.sql

方法2：生成清空所有表的SQL

Mysql –N –s information\_schema –e “SELECT CONCAT(‘TRUNCATE TABLE’,TABLE\_NAME,’;’) FROM TABLES WHERE TABLE\_SCHEMA=’databasename’”

这样就更完善了：

Mysql –N –s information\_schema –e “SELECT CONCAT(‘TRUNCATE TABLE’,TABLE\_NAME,’;’) FROM TABLES WHERE TABLE\_SCHEMA=’databasename’”|mysql databasename

如果有外键的话，可能会报错，因此需要加一个-f参数，强制清空

Mysql –N –s information\_schema –e “SELECT CONCAT(‘TRUNCATE TABLE’,TABLE\_NAME,’;’) FROM TABLES WHERE TABLE\_SCHEMA=’databasename’”|mysql -f databasename