数据库驱动程序测试的建议 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E6_95_B0_E6_8D_AE_E5_BA_93_E9_c104_144422.htm 1. 不要用TestCase的构造函数初始化Fixture，而要用setUp（）和tearDown（）方法。 2. 不要依赖或假定测试运行的顺序，因为JUnit利用Vector保存测试方法。所以不同的平台会按不同的顺序从Vector中取出测试方法。 3. 避免编写有副作用的TestCase.例如：如果随后的测试依赖于某些特定的交易数据，就不要提交交易数据。简单的会滚就可以了。 对于我们来说，有时是必须要提交，以至于有副作用的。 例如：在执行“ 插入 ”后，数据库显然会多出一条数据来。那么必须在随后每个测试自己消除自己的副作用。 在这里，就是自己“ 再删除刚插入的数据 ”。（这时候需要考虑到这个善后的工作不能自己就不能有副作用，删除多了其他的数据）。 这里的副作用还指“ 影响到周围环境 ”，因为我们现在工作的人比较多，所以最好大家的测试服务器能够分开来，例如一个人一个Database实例（可以建得稍微小一点）或者一个人一个数据库， 注意将这些个人之间有区别的内容用常量在每个人自己的所有程序中公用。而不是分布在各个位置。 否则以后要改换测试服务器，所有的程序都需要改动。 为了保证测试程序能够很容易的到处执行，请保证大家的数据库服务器的测试数据全部一致。 否则，就不能做到很容易得拿到FJ也可以很容易的运行，所以需要准备“ 测试数据集 ”。 包括：Schema，table，stored procedure等数据库对象的结构一致， 还包括数据库的数据内容保持一致。 4. 当继承一个测试类时，记得调用父

类的setUp（）和tearDown（）方法。 5. 将测试代码和工作代码放在一起，一边同步编译和更新。（使用Ant中有支持junit的task.） 6. 测试类和测试方法应该有一致的命名方案。如在工作类名前加上test从而形成测试类名。 可能这里我们需要改动，将函数名和我们的测试用例的编号一致起来。 7. 确保测试与时间无关，不要依赖使用过期的数据进行测试。导致在随后的维护过程中很难重现测试。 8. 如果你编写的软件面向国际市场，编写测试时要考虑国际化的因素。不要仅用母语的Locale进行测试。 9. 尽可能地利用JUnit提供地assert/fail方法以及异常处理的方法，可以使代码更为简洁。 这个内容有其关键，assert语句的好坏直接影响到测试的正确性。 因为assert就是用于当前测试项的正确性的。 10.测试要尽可能地小，执行速度快。 1）将所有的数据库的测试数据用ODBC程序自动生成的。 用户可以简单的修改ConnectionString，然后运行程序，就可以创建生成数据库/数据库表/存储结构，并且自动插入数据。 2）为了保证多个测试人员的不干扰，建议分别各自单独使用自己的数据库。否则会因为一个自己的错误，影响别人的工作。 3）在自己的程序中，所有涉及环境的内容都用单独放到一个类中，用static常量共享使用（这样就便于很容易的更换环境再进行测试，做到很容易的移植测试环境）。 4）关于数据库表结构，我建议测试表中含有一个主键，我们在插入数据的时候，保证测试用例，测试用例程序，测试用例程序中的数据，这三者的编号一致起来。便于出现问题时，可以排除数据。 