Rational ClearQuset 配置手册

Jackei.Chan

jackei_chan@hotmail.com
http://blog.csdn.net/jackei/

Rational ClearQuest（以下简称 CQ）是国内目前比较常用的一款缺陷管理工具，它的功能强大，灵活，可实现流程自定义、查询自定义、功能域自定义、用户权限分级管理等功能，并可以集成 Crystal Report 实现更加灵活的报表自定义功能。

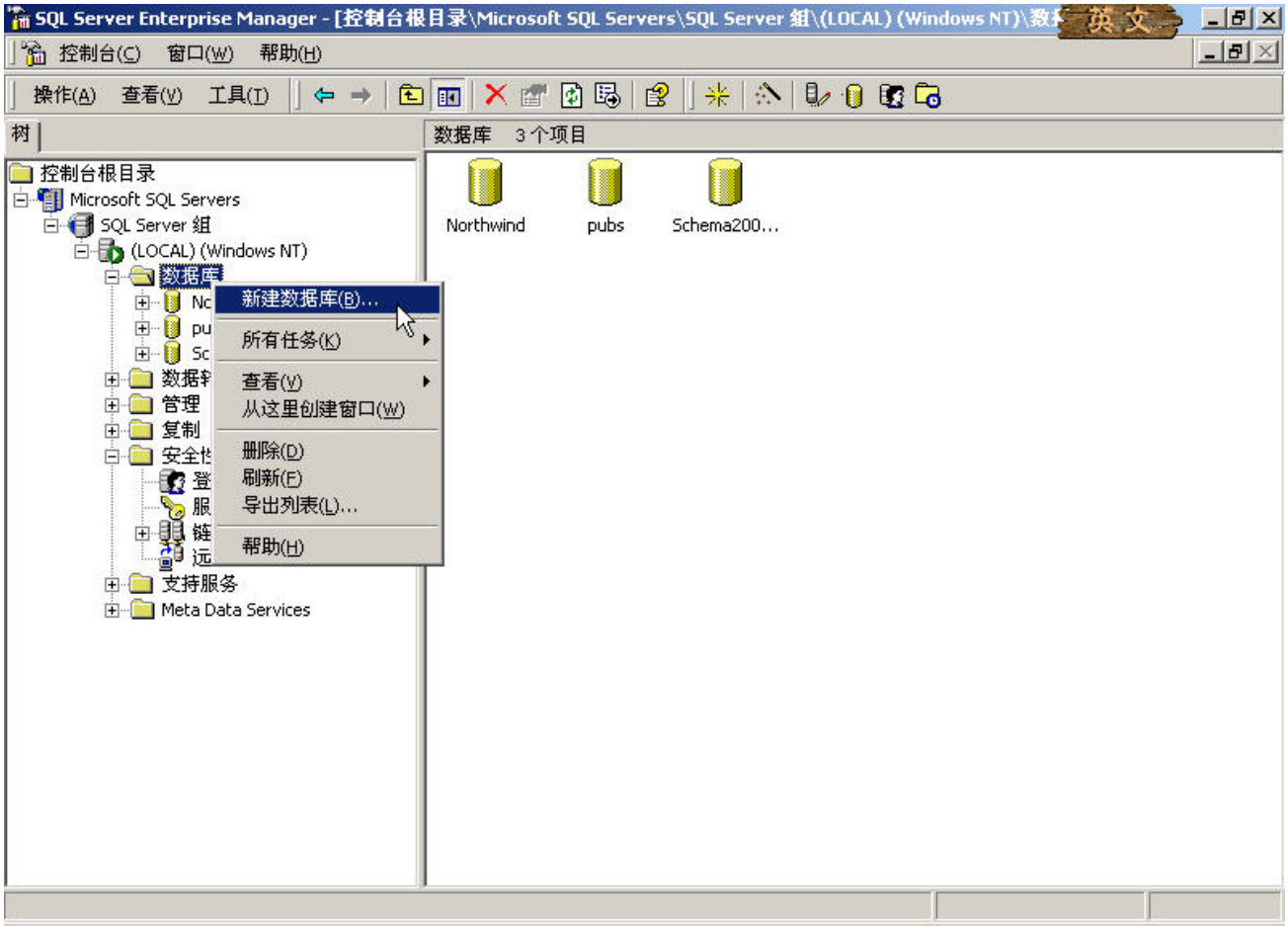
当我们使用 CQ 进行缺陷的跟踪和管理时，所有的信息都是被存放在后台的数据库中的，而我们通过 CQ 提供的操作界面来对数据库中的数据进行操作。CQ 可以支持 Oracle、MS Access、MS SQL Server、IBM DB2、Sybase SQLAnywhere 等常见的 DBMS，下面我们以 MS SQL SERVER 为例，来详细的说明 CQ 的配置过程，所以要求您至少应当熟悉 MS SQL SERVER 2000 新建数据库和登录的操作。

本文中使用的是 Rational ClearQuset 2003，但是请不要向我索要这个软件的 Copy，如有需要，可以同 IBM 公司联系，以便获得免费的试用版。

1. 新增 Master 数据库

Master 数据库是 CQ 的主数据库，用来存放各种预先定义的流程模板、功能域模板等数据。新建 SQL Server 数据库需要使用 SQL Server 的“企业管理器”，如下图所示

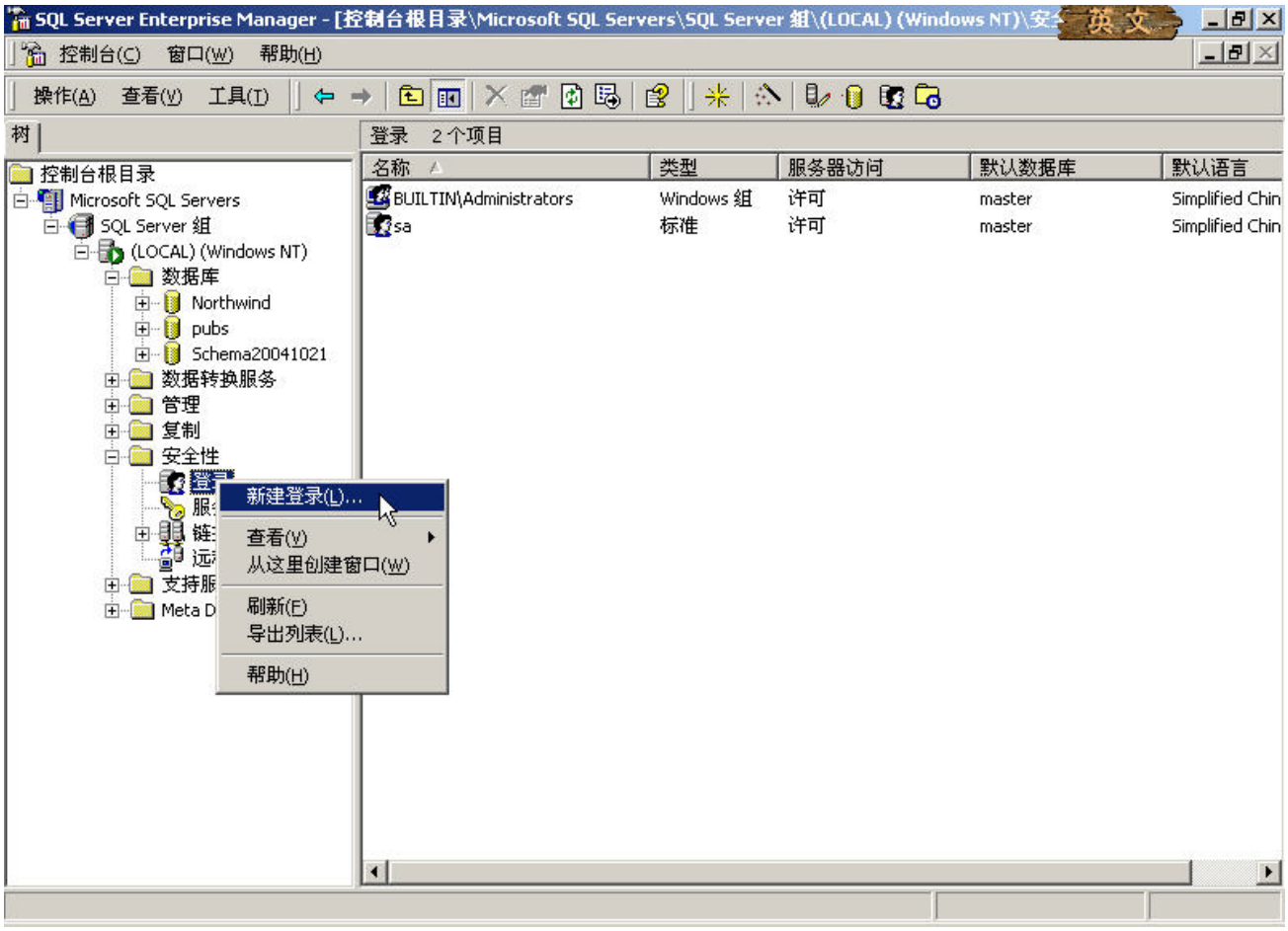
打开企业管理器后，使用“新建数据库”功能来新建一个空的数据库，如下图所示



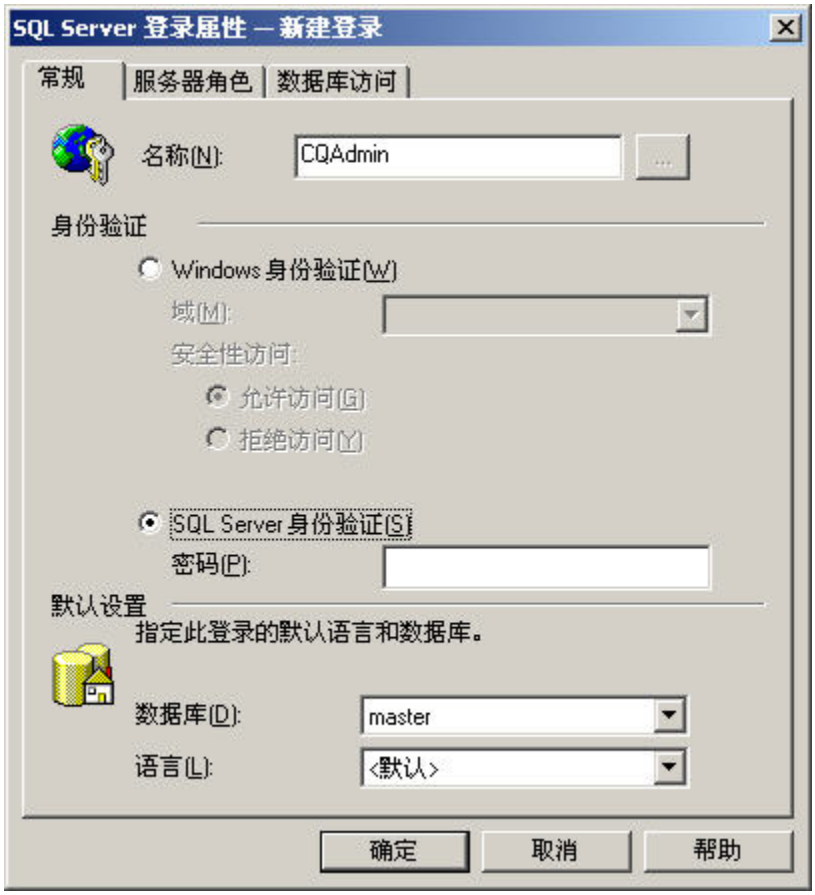
然后输入“数据库属性”页面中的“名称”项，点击确定，如下图

注意：在新建数据库时一定要使用 SQL Server 的用户，而不是 windows 用户登录。但是不要使用 SQL Server 默认的 SA 用户，因为使用该用户会对将来进行数据迁移造成影响，可以在 SQL Server 中新增一个用户，并使用这个新建用户重新连接到 SQL SERVER 来进行这项操作。

新增数据库后，我们还需要为 CQ 添加一个新的数据库登录用户，参见下图

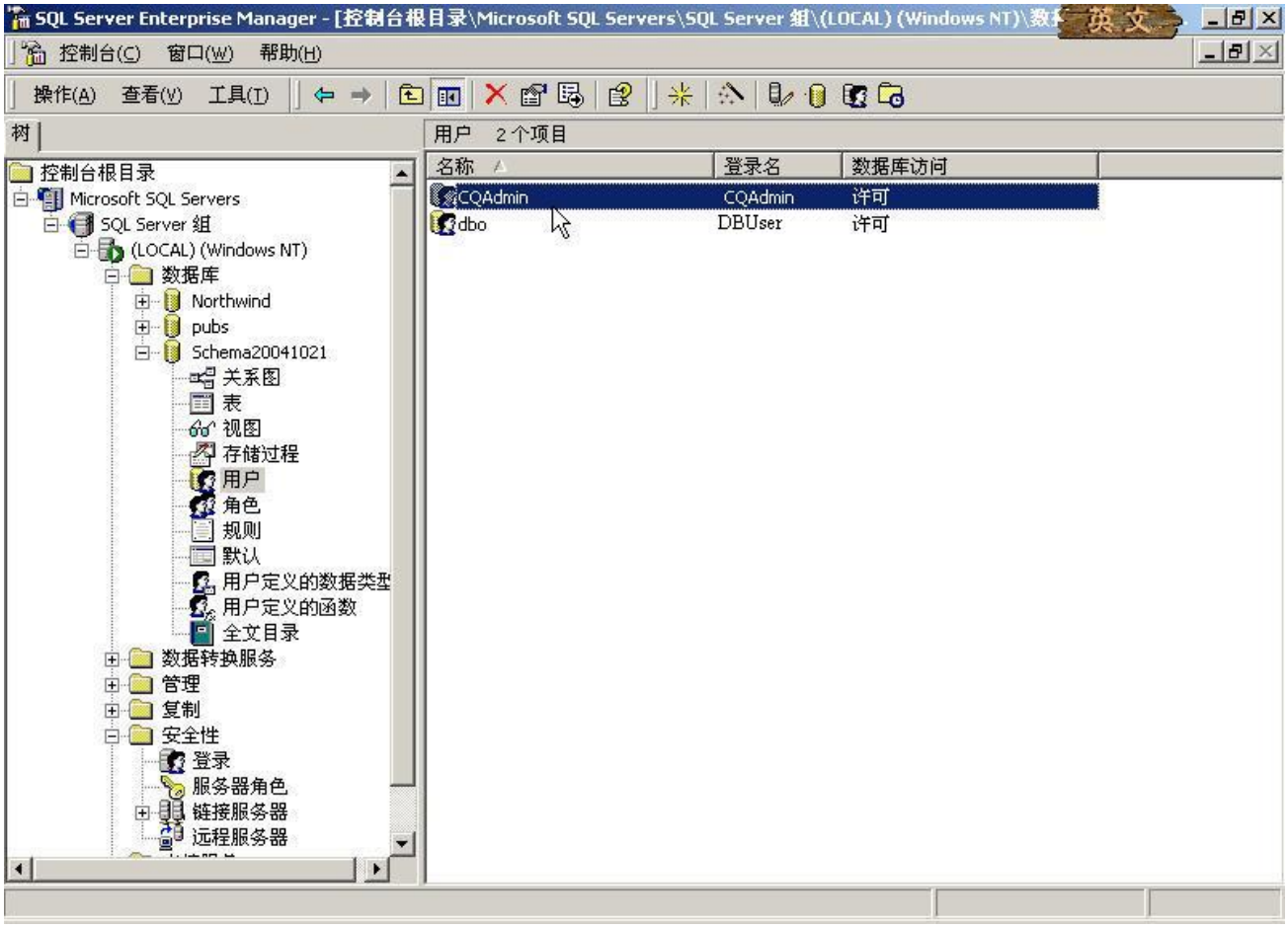


这里一定要注意登录用户属性的设置。对于“名称”可以自定义一个比较容易记忆和理解的英文字符串，比如“CQAdmin”，“身份验证”选择“SQL Server 身份验证”，密码可以留空。



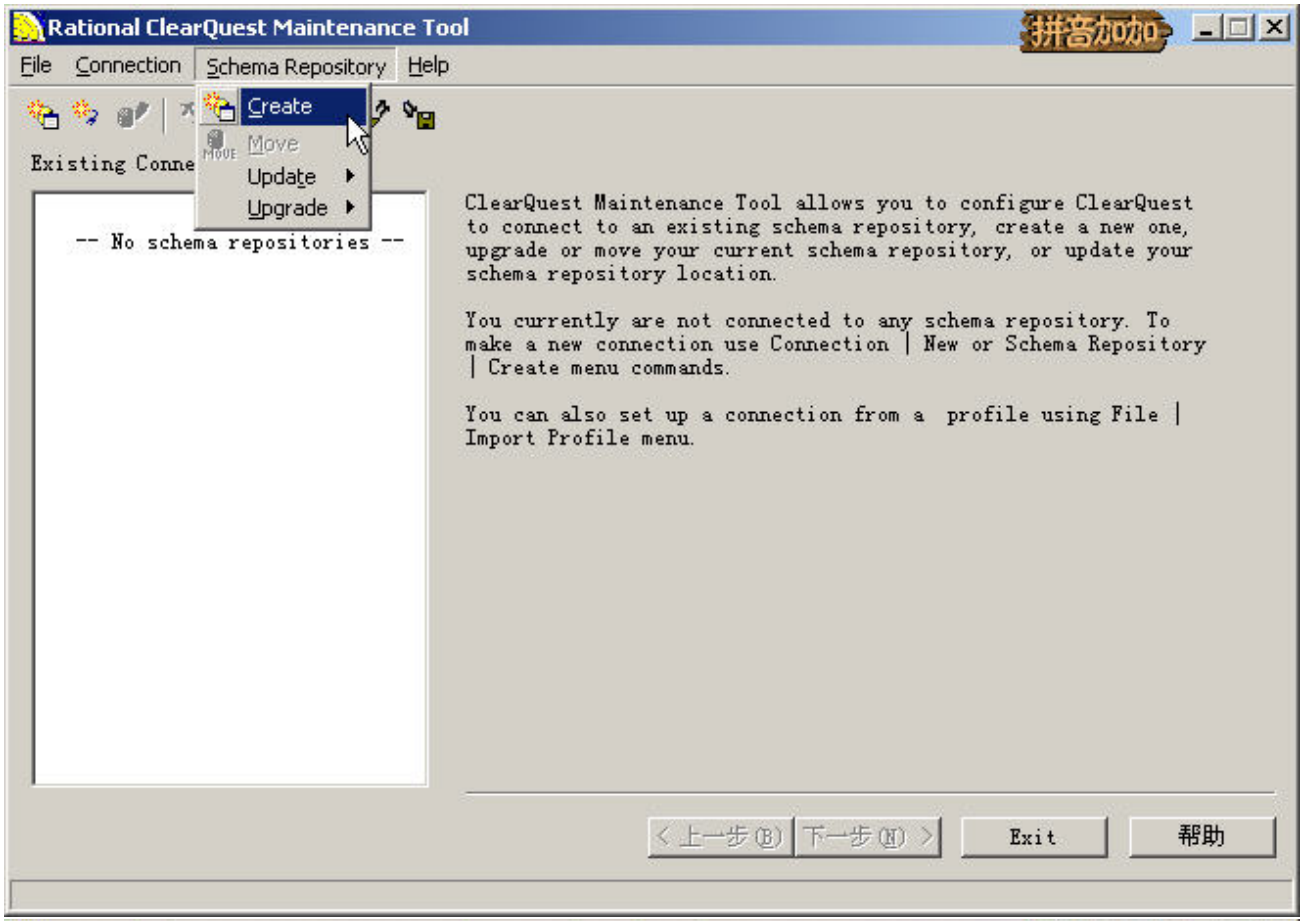
在登录用户属性的“数据库访问”页面中，选定刚刚新建的数据库，并赋予 public 和 db_owner 的角色权限，如下图

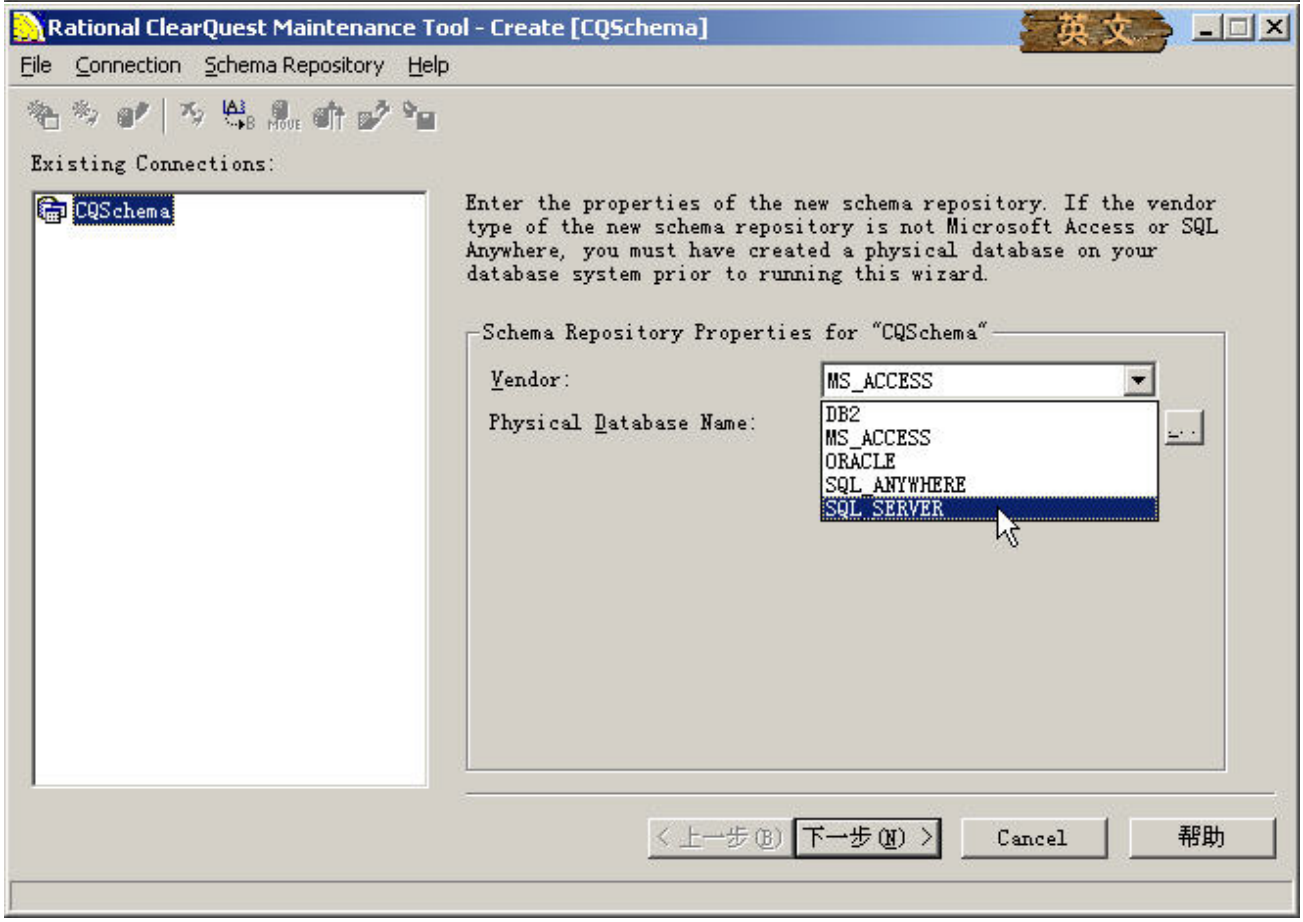
添加完毕之后，可以在新增的“Schema20041021”数据库的“用户”项中看到刚刚新增的登录用户，自此完成 SQL Server 数据库的建立。

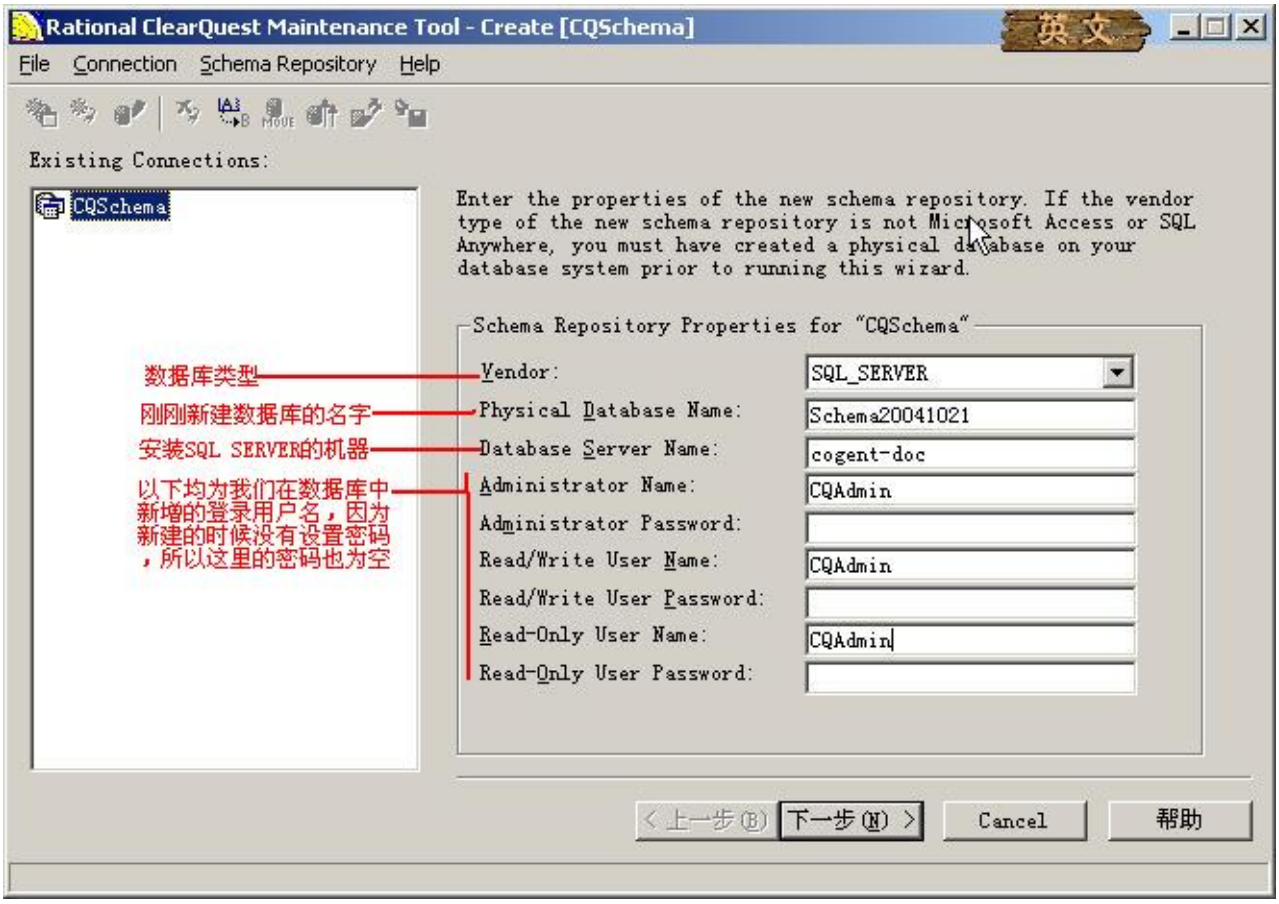


2. 新建 Schema Repository

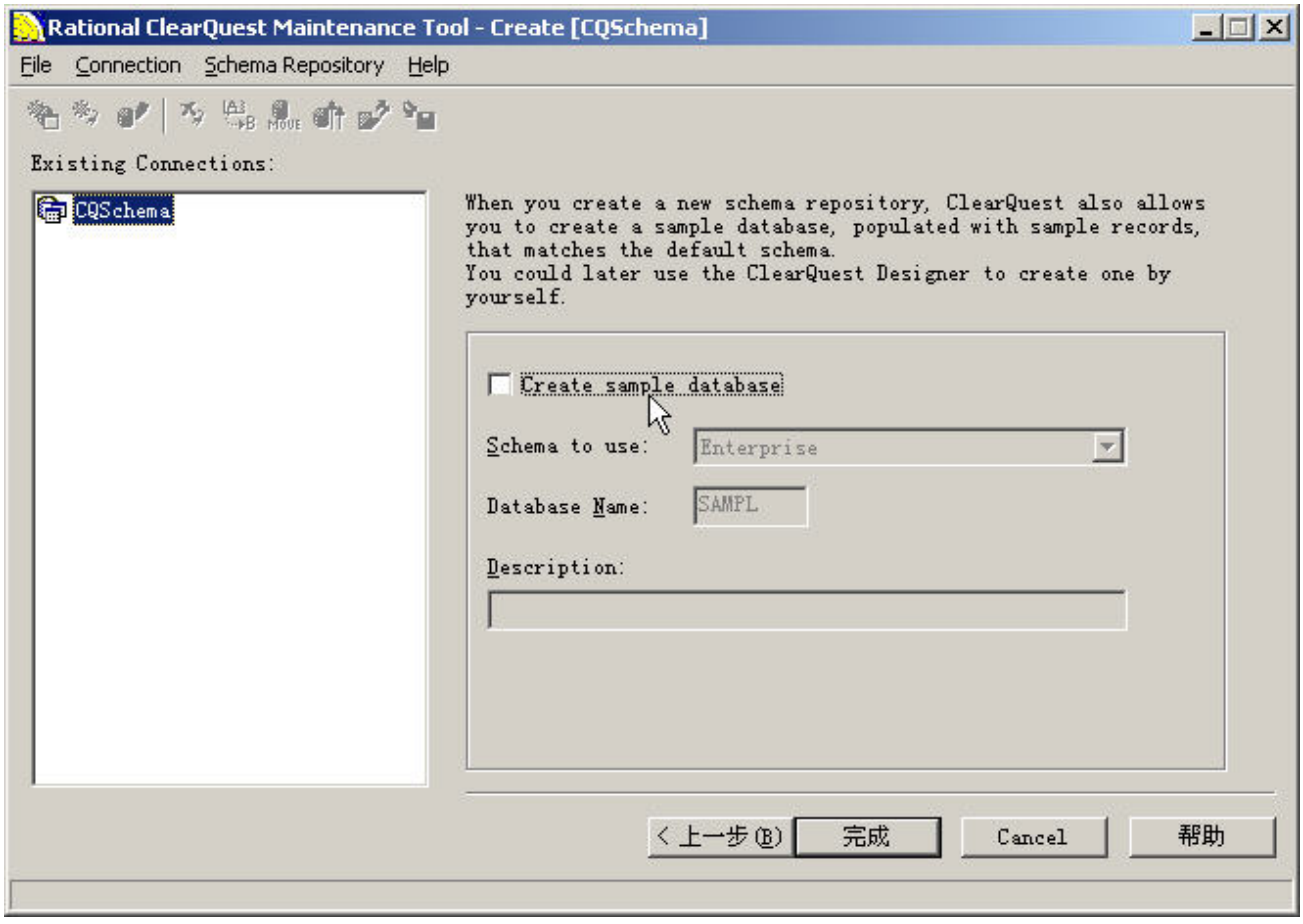
打开 ClearQuest Maintenance Tool 工具，选择“Schema Repository”菜单下的“Create”项，新建一个 Schema，然后在“Vendor”下拉列表中选择“SQL SERVER”项。并填写物理数据库名、数据库服务器名、管理用户和读写用户名，然后点击“下一步”按钮，如下图所示


Rational ClearQuest Maintenance Tool
File Connection Schema Repository Help
Create
Move
Update
Upgrade
Existing Conne
-- No schema repositories --
ClearQuest Maintenance Tool allows you to configure ClearQuest to connect to an existing schema repository, create a new one, upgrade or move your current schema repository, or update your schema repository location.
You currently are not connected to any schema repository. To make a new connection use Connection | New or Schema Repository | Create menu commands.
You can also set up a connection from a profile using File | Import Profile menu.
< 上一步(B)
下一步(N) >
Exit
帮助

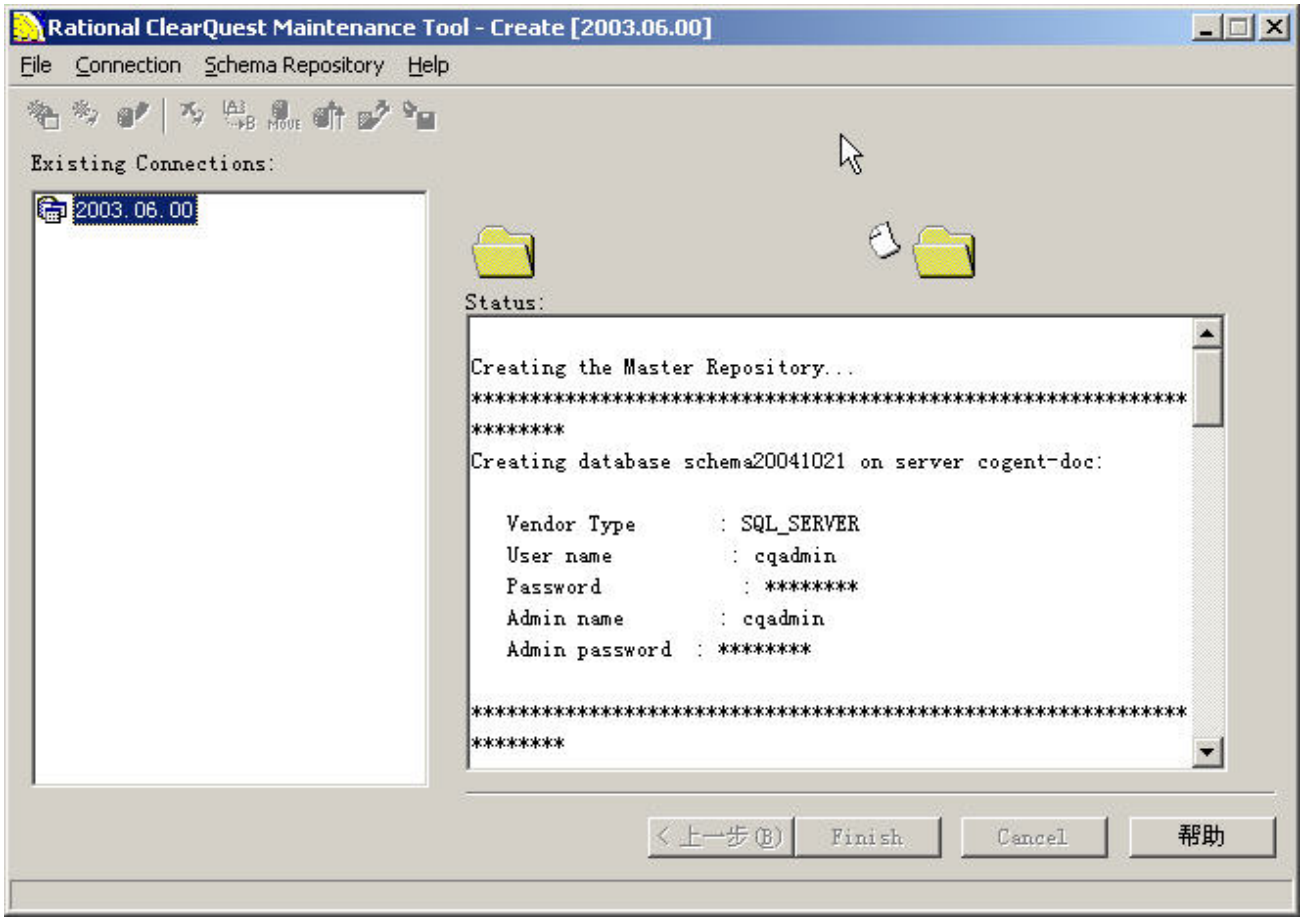

Rational ClearQuest Maintenance Tool - Create [CQSchema]
File Connection Schema Repository Help
Existing Connections:
CQSchema
Enter the properties of the new schema repository. If the vendor type of the new schema repository is not Microsoft Access or SQL Anywhere, you must have created a physical database on your database system prior to running this wizard.
Schema Repository Properties for "CQSchema"
Vendor:
MS_ACCESS
DB2
MS_ACCESS
ORACLE
SQL_ANYWHERE
SQL_SERVER
Physical Database Name:
< 上一步(B)
下一步(N) >
Cancel
帮助



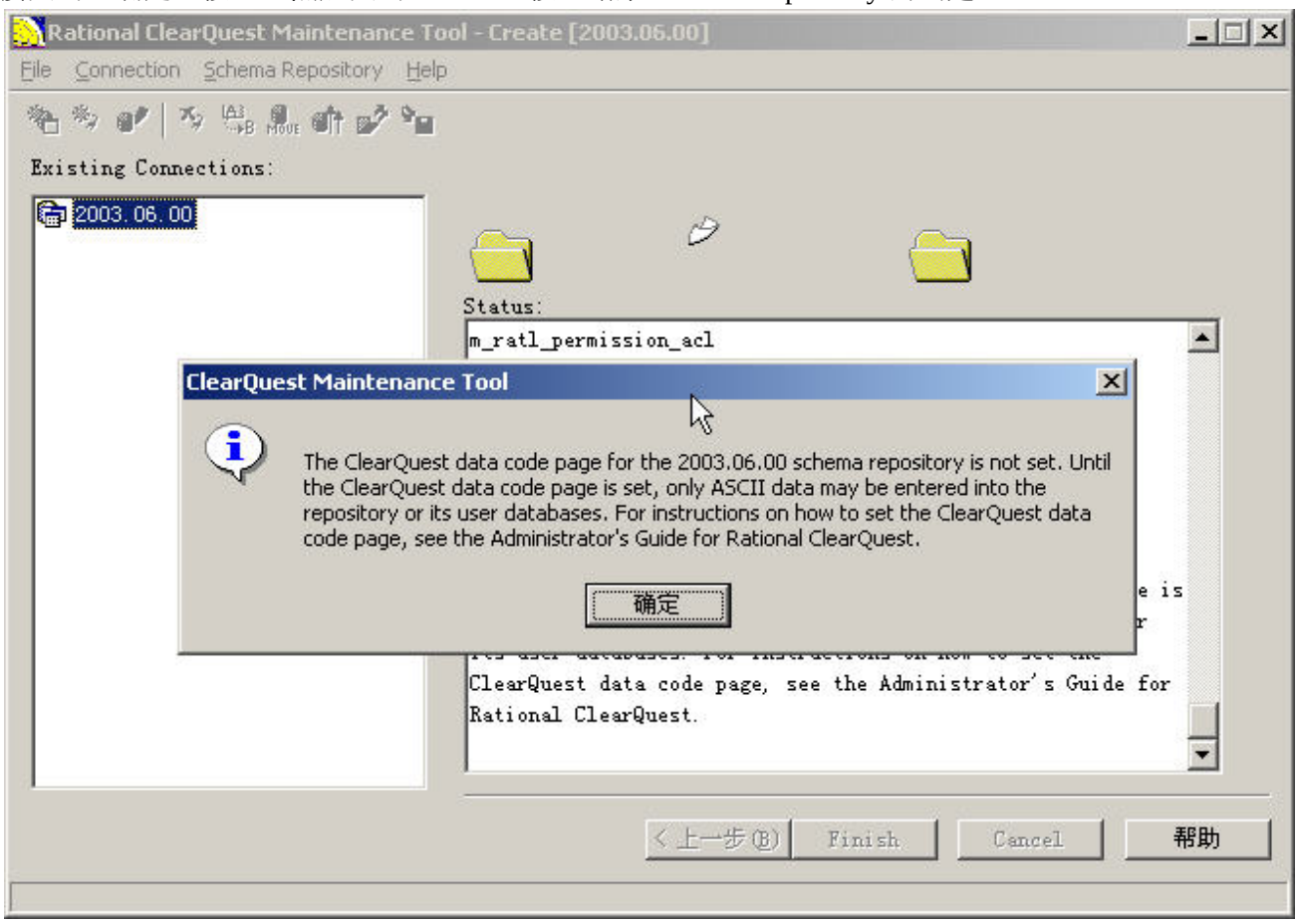
然后取消掉“Create sample database”项，点击“完成”按钮

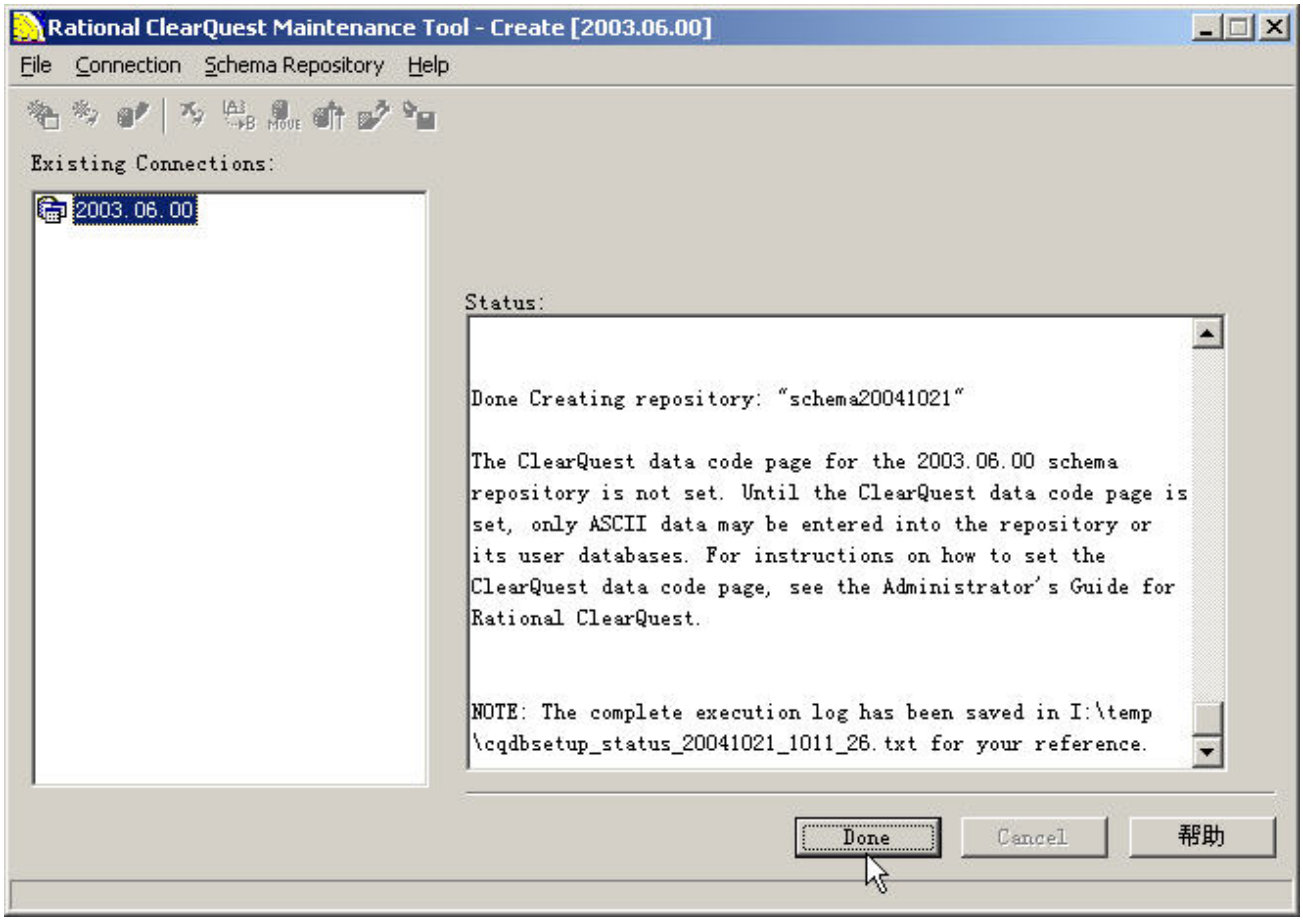


之后系统会自动在数据库中创建 CQ Schema 初始化数据



在创建结束时，系统会出现一个提示信息（我们暂时可以先不管它，在后面会讲到如何处理这个问题），直接点击“确定”按钮，然后点击“Done”按钮结束 Schema Repository 的创建。





注意：在进行上述操作时，需要先关闭诺顿防火墙。

3. 修改 CQ 管理员密码

打开“ClearQuest Designer”，使用默认管理用户登录（用户名为 admin，密码为空），不必理会弹出的提示信息（等会我们会处理它），在弹出的 Schema 选择列表中点击“取消”按钮。然后选择 ClearQuest Designer 工具栏中的“User Administrator”按钮，然后在用户列表中选择“admin”用户，并修改 admin 用户的密码（比如我们在这里改为“tester”），在填写“Password”和“Confirm Password”项后点击“OK”按钮退出当前界面。接着点击“User Administrator”界面的“OK”按钮退出时，系统会提示是否要保存刚刚修改的信息，点击“是”退出“User Administrator”界面，并退出 ClearQuest Designer 工具。

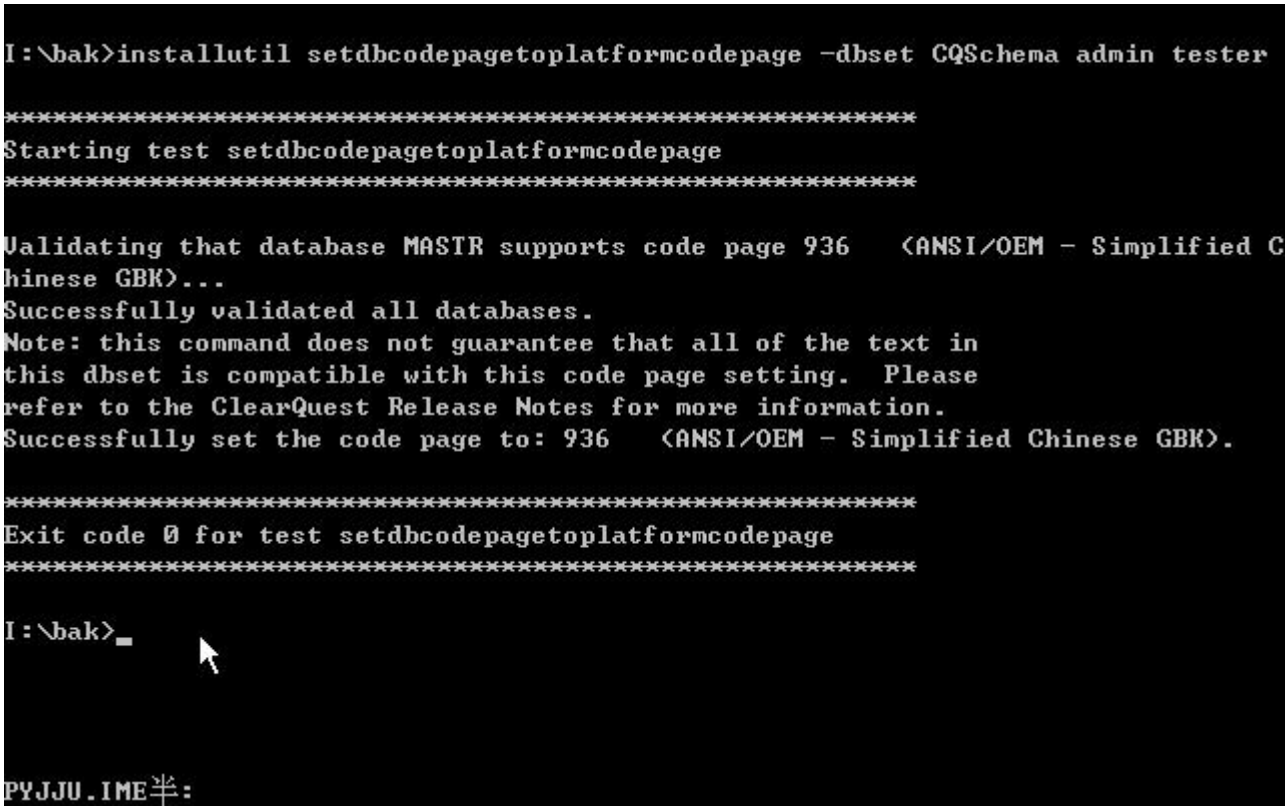
4. 配置 code page

code page 是在 Rational 2003 中新增的一个特性，在 2002 中是不用费劲搞这个东东的。设置的方法非常简单，只要在命令行方式下输入下面的命令，执行通过就可以了。

installutil setdbcodepagetoplatformcodepage -dbset CQSchema admin tester

其中“installutil setdbcodepagetoplatformcodepage -dbset”是命令和参数，“CQSchema”是我们在 ClearQuest Maintenance Tool 中建立的 Schema 的名字，而“admin”和“tester”分别是 CQ Schema 的管理员用户名和密码。**（注意，这里一定要使用管理员来进行设置，并且密码一定不可以为空，这也是上面我们要修改管理员密码的原因。）**

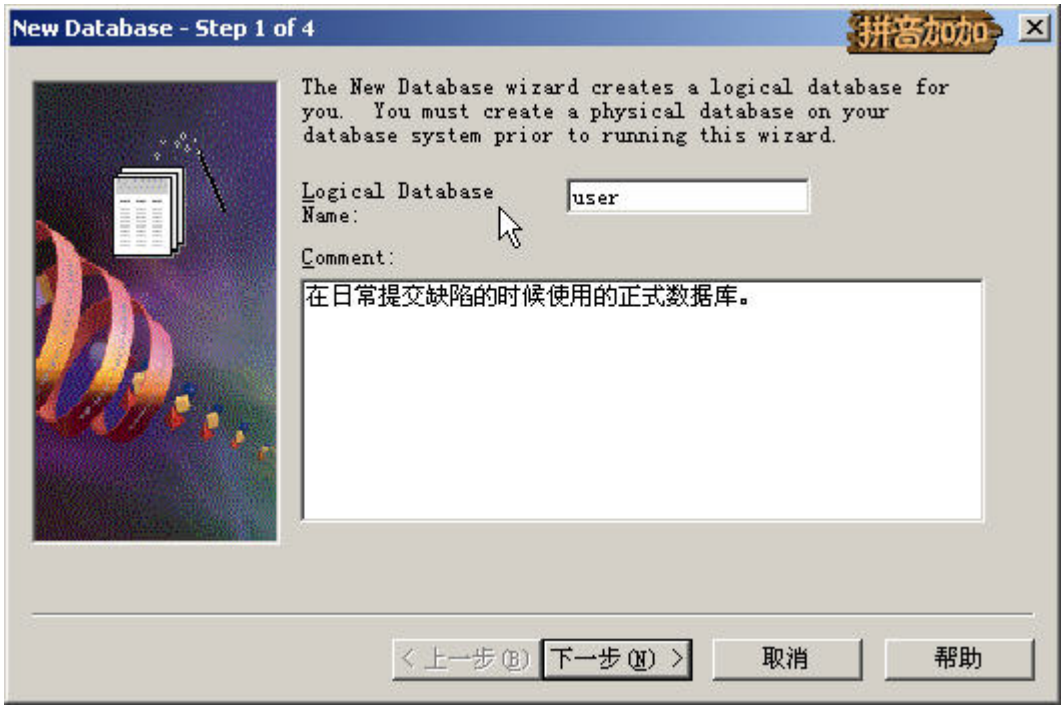
执行通过后，会显示类似下面的信息，如果没有看到错误提示，那么就说明已经配置成功了。再次登录 ClearQuest Designer 的时候就不会有提示设置 code page 的信息了。



5. 添加 Production Database 和 Test Database

Production Database 和 Test Database 是使用 CQ 是必须建立的另外两个数据库，加上前面的 Master 数据库就一共是三个数据库了。对于 Production Database 和 Test Database 要使用同 Master 数据库同样的方法先在 SQL SERVER 上新建两个空的数据库（本例中新建了 ProducationDB20041021 和 TestDB20041021 两个数据库。Production Database 和 Test Database 同 Master 数据库分别保存了同 CQ 有关的不同的信息。对于 Master 数据库，它保存了 CQ 预设的一些方案（Schema），包括预设的处理流程、用户界面、数据库字段等信息；而在 Production Database 和 Test Database 中保存的是某个具体的 Schema 进行个性化配置后的信息，另外，还包括我们提交的所有缺陷以及同缺陷处理相关的所有信息。这在下面的介绍中会通过实际操作来进行更深入的讲解。）

在新增了两个空的数据库之后，我们还要在 CQ Designer 中建立 CQ 同这两个数据库的关联。如果已经在 ClearQuest Maintenance Tool 建立了多个 Schema 或者 Schema 链接，则在登录 CQ Designer 的时候，会出现一个 Schema 列表让你选择，我们可以选择上面建立的那个叫 CQSchema 的 Schema（如果只有一个 Schema，就不会出现选择列表，直接默认登录这个 Schema）。然后选择 CQ Designer 界面中的 Database 菜单下的“New Database”菜单项，来建立 CQ 同一个新的数据库的关联。如下图所示，



我们先建立一个 Producation Database，在 Logical Database Name 项输入一个同 SQL Server 中的数据库相对应的逻辑数据库的名字（也就是 SQL Server 中的数据库在 CQ 中所使用的别名），下面的“Comment”中可以添加对于这个数据库的描述，比如是用来保存哪个项目的缺陷记录。这里注意一下，Logical Database Name 项是有输入长度限制的，只能输入 5 位数字或英文（为什么这样设计，我现在也没有弄明白），我们在命名的时候应当使用有意义的名字。

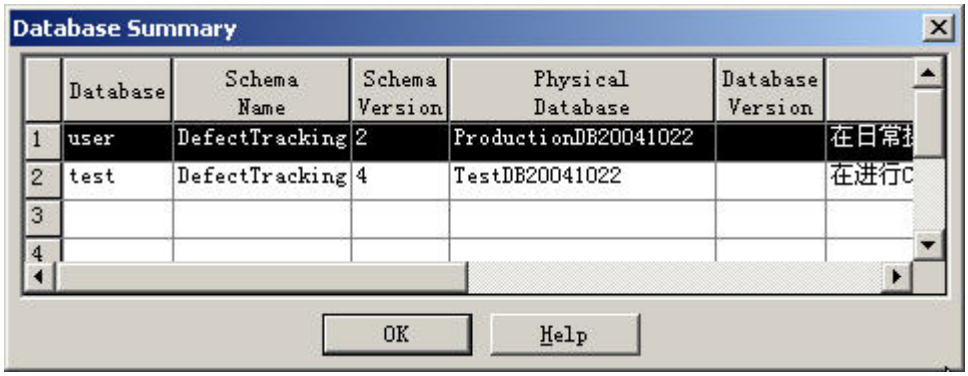
接下来需要填写数据库链接信息，方法同新建 Schema 的时候是一样的。

不过一定要注意，要选择“Producation Database”项，这样才能将这个数据库标志为 Producation Database 数据库。点击“下一步”按钮后是进行超时设置，使用默认设置就可以了，然后继续“下一步”。下面这一步也是至关重要的一步，我们需要把 Producation Database 同 Master 中存储的多个 Schema 中的某一个进行关联，之后这个被关联的 Schema 的信息就被系统完全复制到 Producation Database 中，供我们根据自己的需要进行调整，如下图。

我们在这里选择的是“DefectTracking”，这是一个专门用来进行缺陷跟踪的 Schema。这个列表中的 Schem 都是在 ClearQuest Maintenance Tool 中建立 Schema Repository 时由系统自动添加到 Master 数据库中的。我们可以通过 Schema Name 来看出不同的 Schema 所应用的环境是不同的，这是因为 CQ 不仅仅是只能应用在缺陷管理方面。继续点击“完成”按钮，稍等片刻后系统会提示操作成功。

之后我们再使用同样的方法建立 Test Database，不过在第二个步骤要选择“Test Database”项，如下图所示

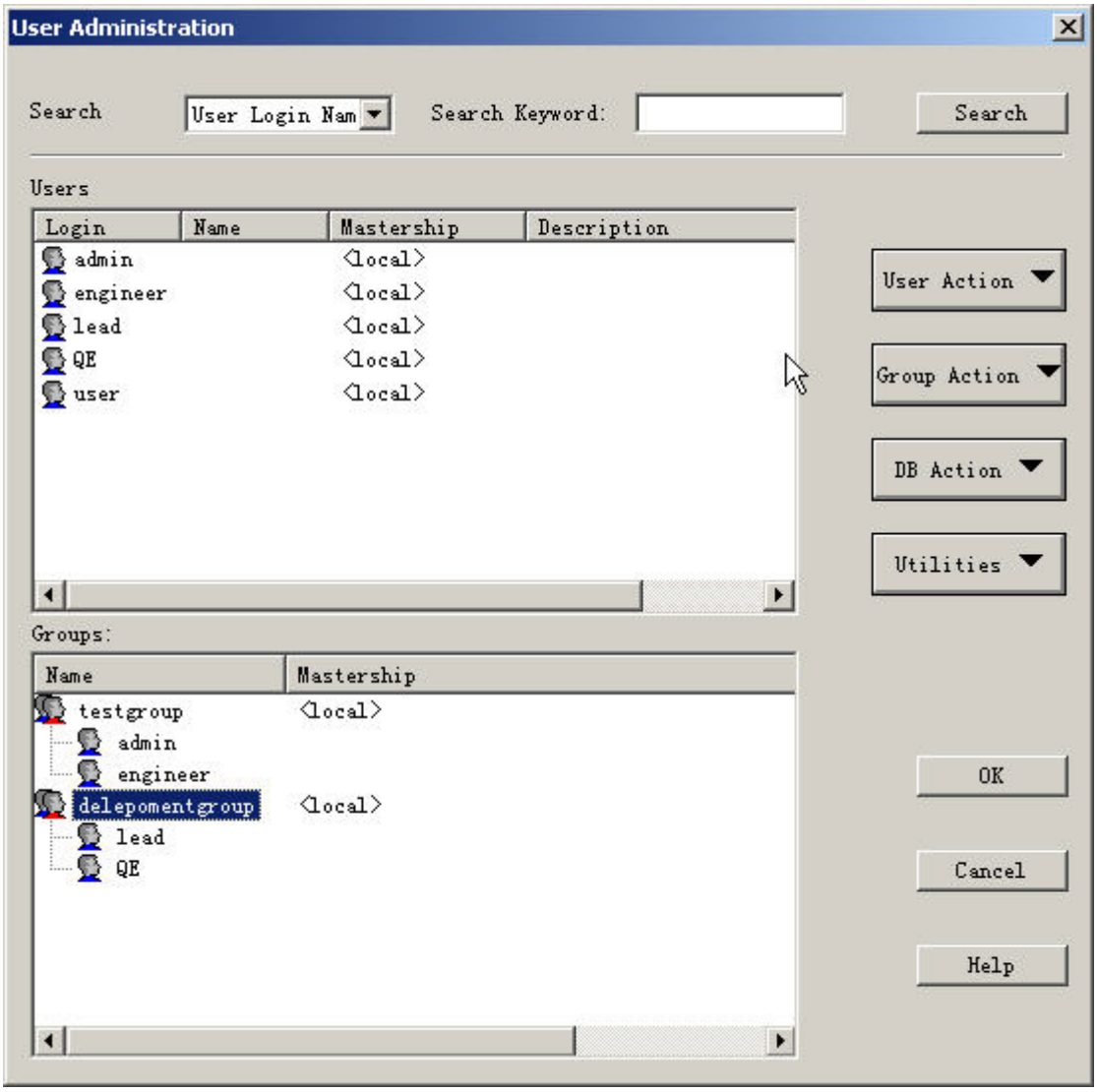
在上面的操作完成之后，我们可以在 CQ Designer 中选择 View 菜单下的“Database Summary”项来查看一下结果，如下图所示



其中“Database”字段中是我们在建立 Producation Database 和 Test Database 时填写的 Logical Database Name，以后我们在 CQ 中进行操作时，看到的都是这个名字；“Schema Name”字段中显示的是这两个数据库关联的 Schema 的名字；“Schema Version”字段显示的是这个数据库中保存的 Schema 的版本，这个问题稍后我们将再详细讲解；“Physical Database”字段中显示的是我们的 Producation Database 和 Test Database 所对应的 SQL Server 的数据库名，是使用 SQL Server 企业管理器建立的实实在在的数据库。

6. 用户管理

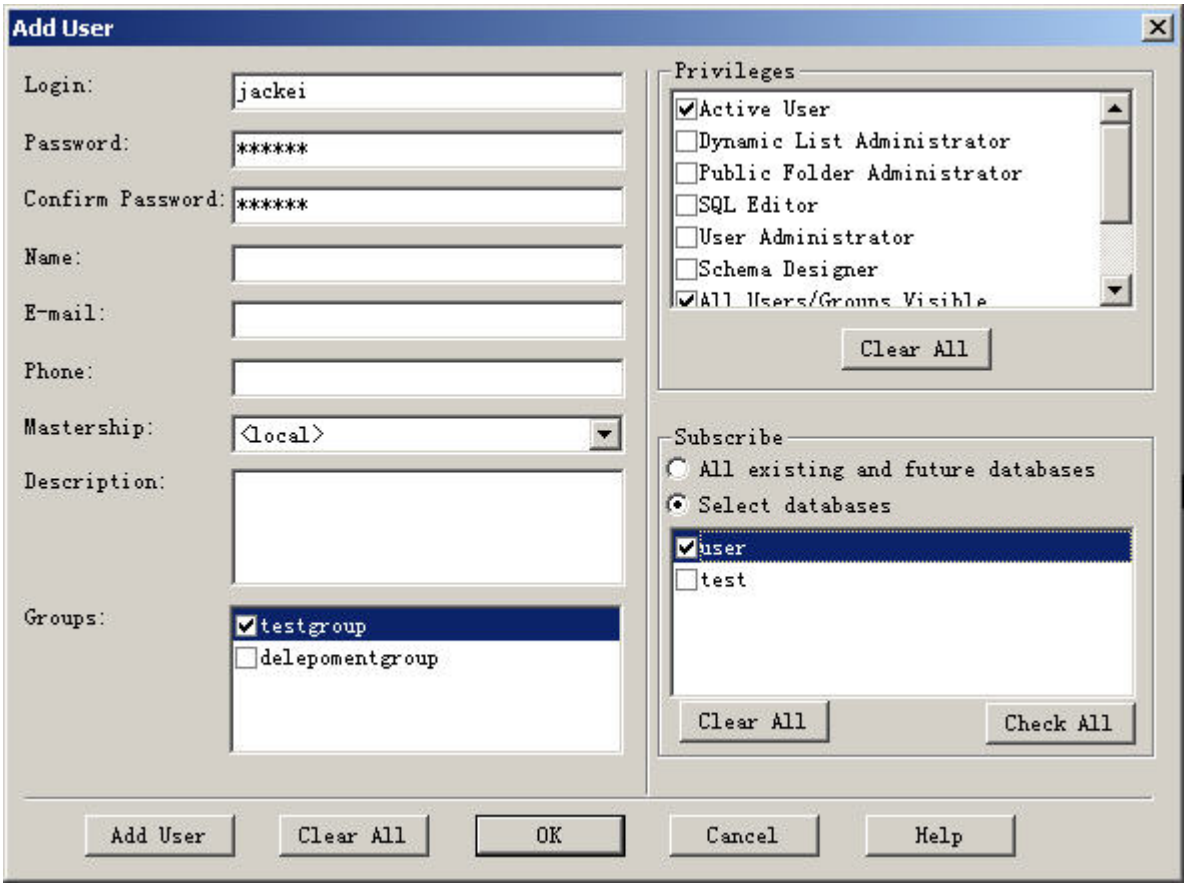
首先点击 CQ Designer 工具栏中的最后一个按钮——User Administrator——进入用户管理界面。如下图所示



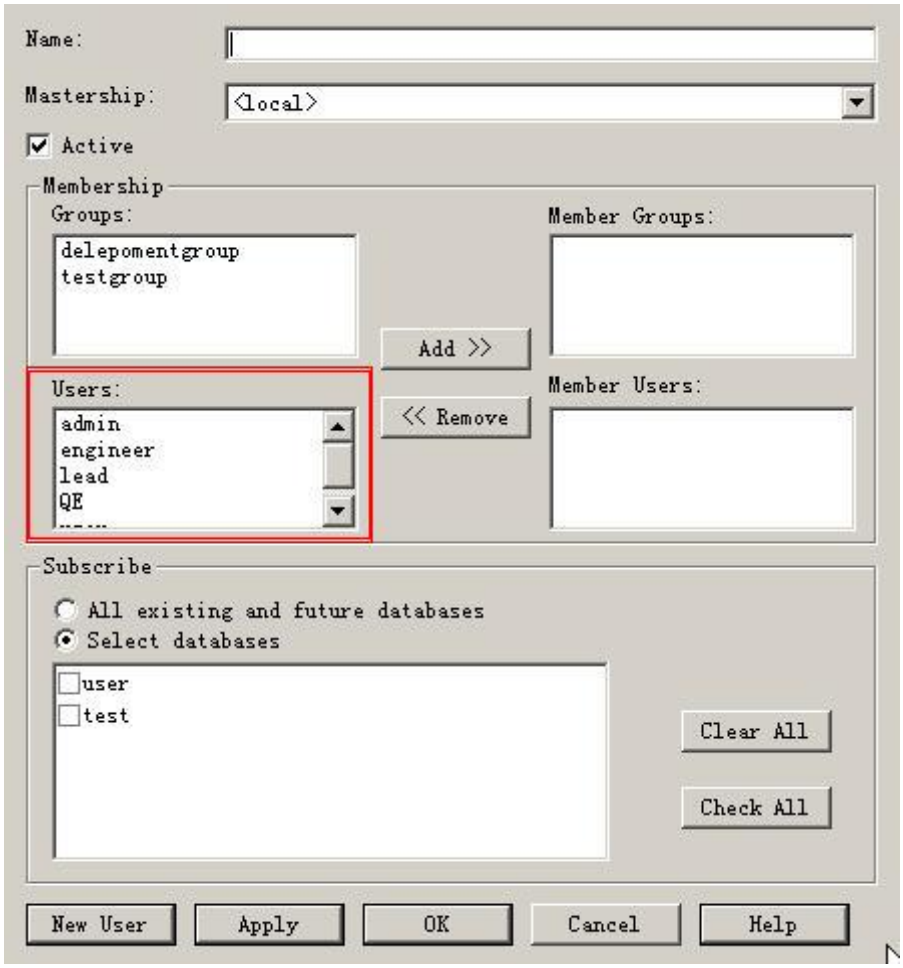
CQ 提供了灵活的用户管理方式，你可以通过设置用户所关联的组、数据库以及权限，来赋予某个用户

所能行使的权限。下面我们通过一个新增用户的例子来实际看一下 CQ 中的用户管理方式。

点击用户管理界面右侧的“User Action”按钮，选择“Add User”项，然后添加“login（登录 CQ 使用的用户名）”和密码，然后选择相应的组，这里选择的是“TestGroup”，再选择数据库，这里选择的是 user 数据库（注意，这里的 user 就是我们上面所说的 Logical Database Name）。当然，也可以选择全部的数据库。不过我们这里没有选择 test 数据库，也就是说如果需要登录到 test 数据库进行操作，则当前新增的用户是没有任何权限的。另外，还可以对“Privileges”项进行设置，来改变当前用户在已选定的数据库中的权限（但是通常我们建议除了管理员，其他用户均使用系统默认的权限），如下图所示



我们在添加用户是使用的 Groups 信息（比如上面的 testgroup、developmentgroup）并不是 CQ 预设的，而是我们通过 CQ 的分组维护功能添加的。在实际操作时，我们可以通过点击 user Administrator 界面右侧的 Group Action 按钮，选择 Add Group 项，这时系统会弹出 Add Group 界面来等待输入，如下图所示

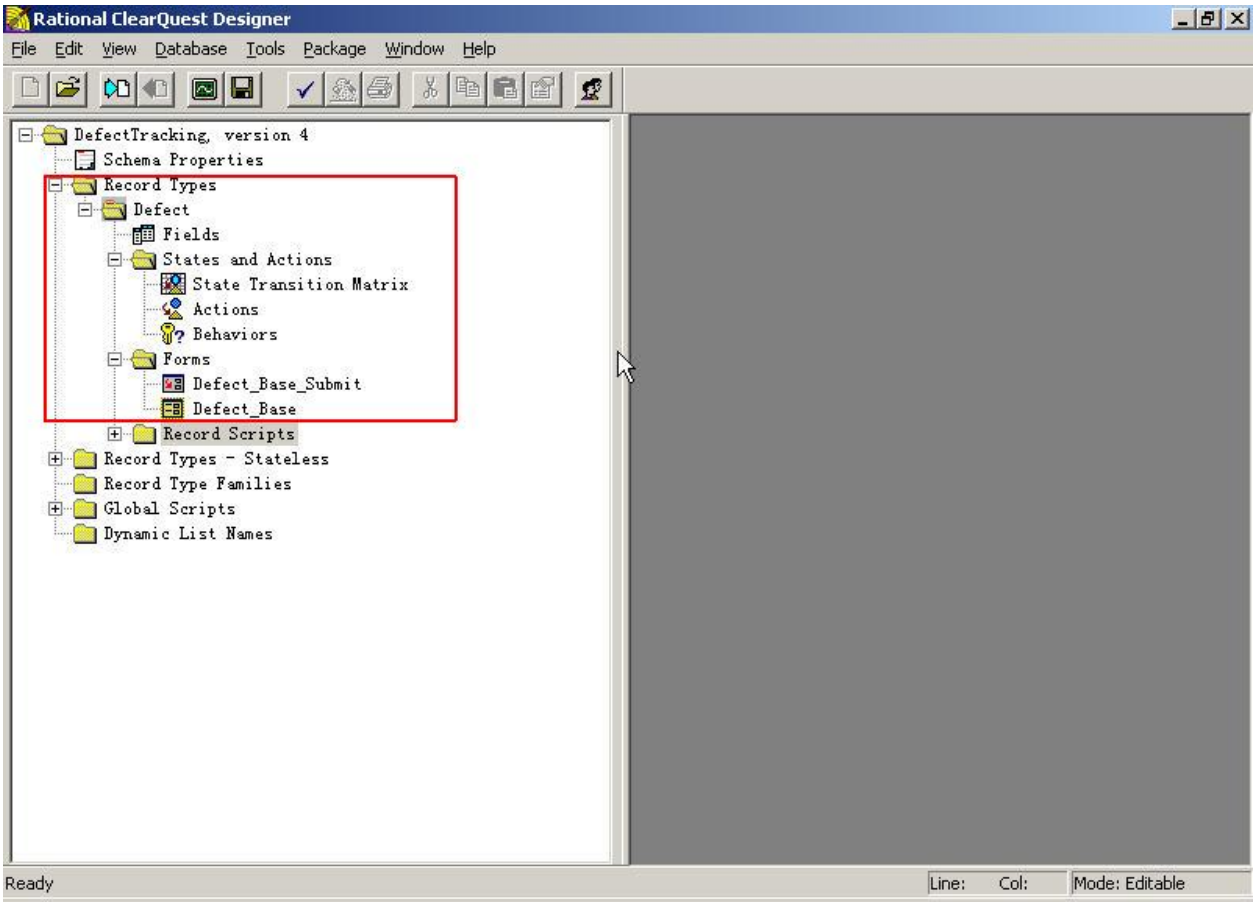


我们首先需要输入 Name（在 CQ 中显示的组的名字），然后从 Users 列表中选择需要加入这个组的用户，点击 Add 按钮，将这个用户添加为 Member Users。我们可以看到在 Add Group 界面的下方也有一个选择数据库的功能，那么这里同新增用户那里的选择数据库有什么区别呢？其实 CQ 中对于用户权限的管理是分级的，某个组的所有成员，都具有这个组的权限，但同时，组中的每个用户也可以拥有自己不同于其他成员的权限。对于 CQ 中权限控制的原理，将会在本文的最后会做专门的讲解。

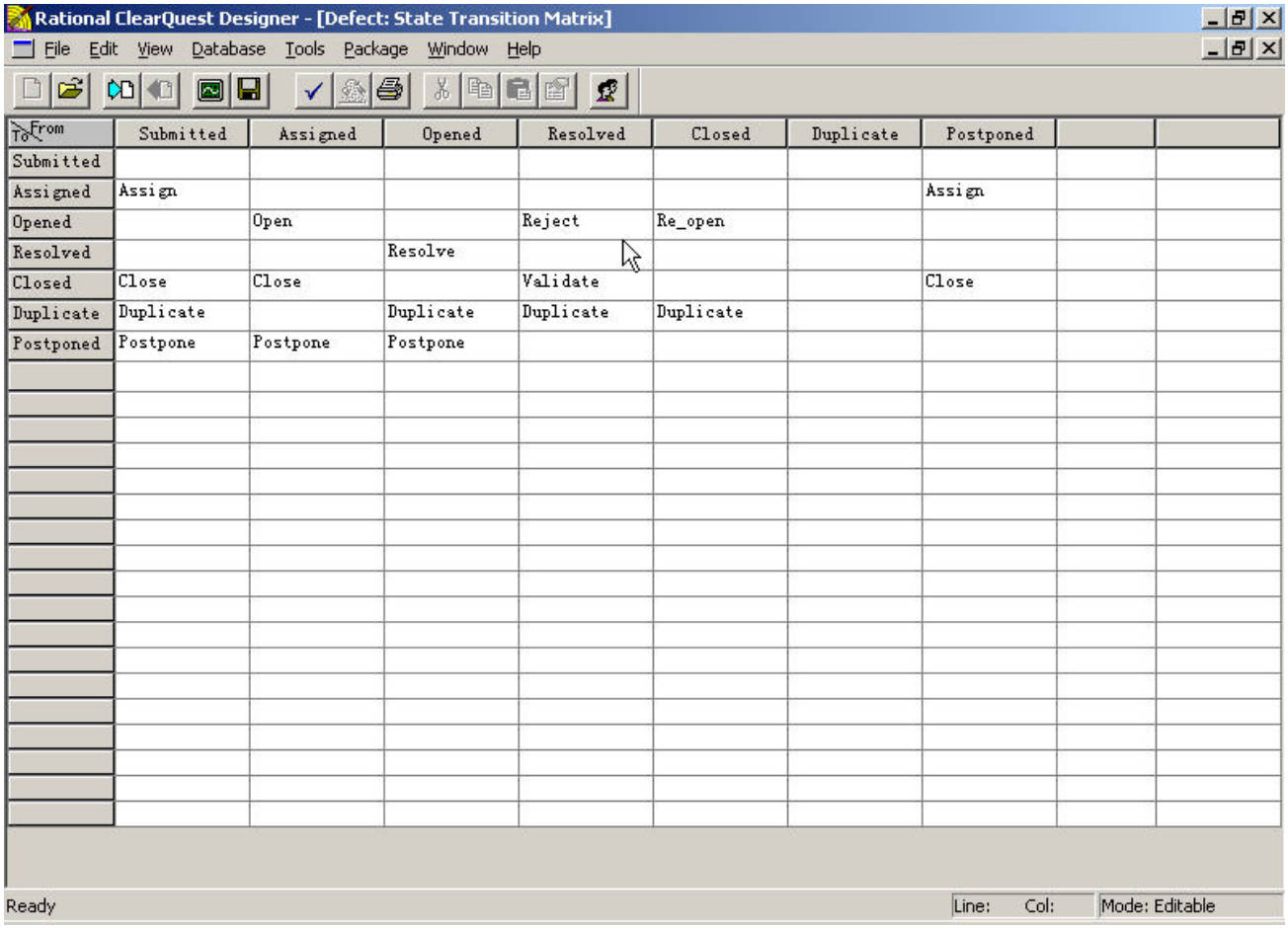
另外，因为提交的缺陷记录中会记录提交者，所以 CQ 中没有提供删除用户的功能，如果有某个用户被废弃，则只能通过取消权限的方式来禁止登录到 CQ。

7. 定制缺陷管理流程

虽然我们选择的 Schema 中也提供了预设的缺陷管理流程，但是在实际工作中，这个流程常常不能完全符合我们的工作要求，所以我们会使用 CQ Designer 来重新定制或调整已有的流程。如下图所示。目前我们所需要操作的内容都位于 Record Type 下的 Defect 中，而流程定制需要使用的是“State Transition Matrix”和“Actions”两项。



我们先来看一下“State Transition Matrix”中的内容，如下图所示



就如上面的名字所示，这是一个“状态转换矩阵”。

这里要先说一下状态的概念。我们在工作中提交的缺陷，都是已记录的形式保存在数据库相应表内的记录。在缺陷跟踪管理的过程中，随着不同的阶段，我们会将缺陷记录修改为不同的状态来进行标记。在上面的矩阵中，上方和左侧的灰色区域是缺陷的状态（上方的是“源状态”，左侧的是“目标状态”），而之间的白色区域则是进行状态（State）转换所需要进行的操作（Action）。 例如，上方的第一个状态 Submitted 是缺陷刚刚被提交的时候由系统自动赋予的状态；当项目经理通过 CQ 看到缺陷，并将它分配给某个相应的负责人，这时缺陷的状态变为 Assigned；而当相应的负责人看到分配给自己的缺陷，比如程序员，开始修改的时候，会把缺陷标志为 Opened，在最终修改结束并提交回归测试的时候，会再次标志为 Resolved；而当测试人员发现有 Resolved 的缺陷时，则进行相应的回归测试，如果回归测试通过，则标志为 Validated，否则使用 Reject 操作重新标志为 Opened 状态，由缺陷负责人重新检查。其他的状态和操作的含义各有不同，不过道理都是一样的。

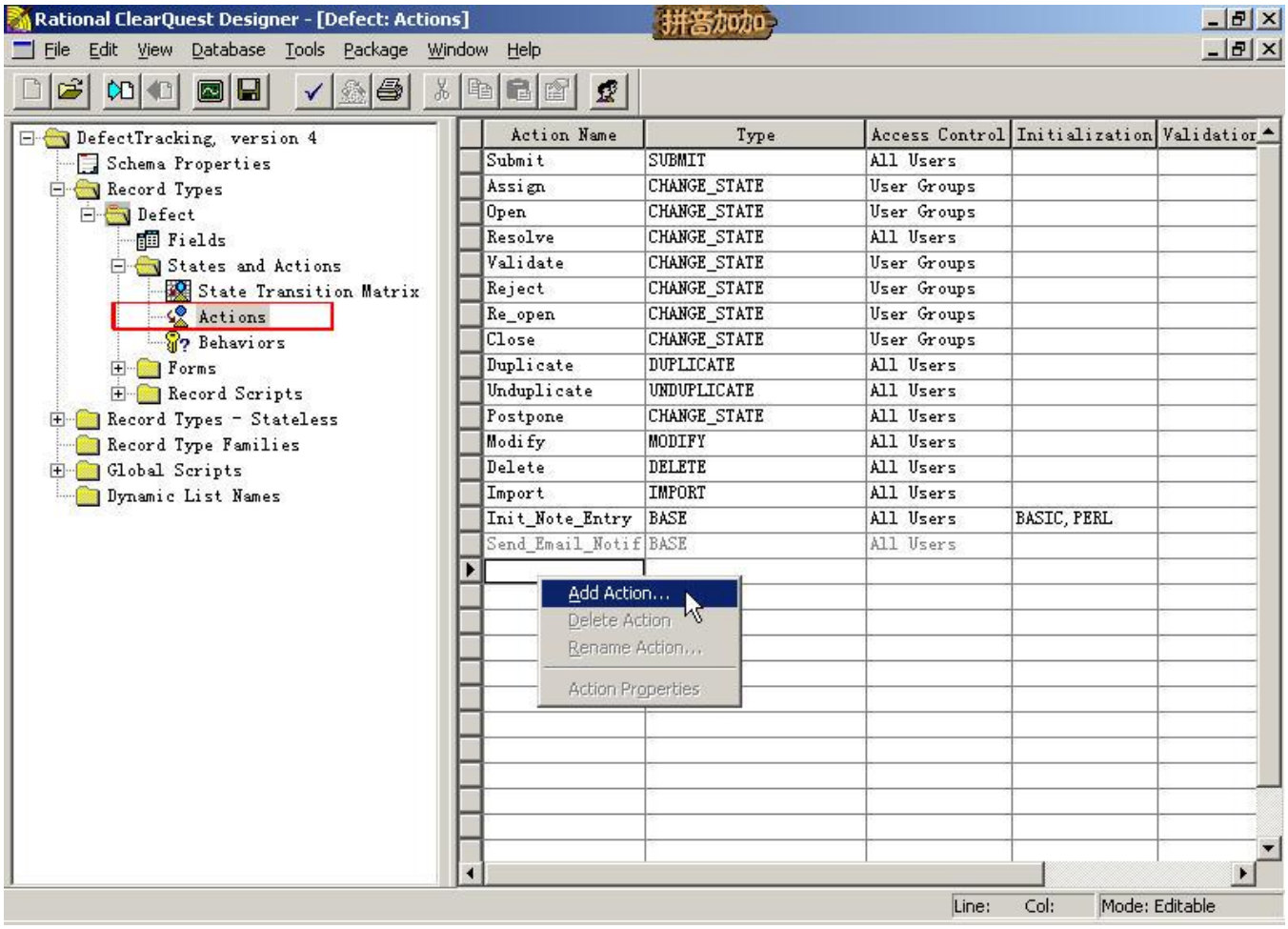
如果希望添加新的处理流程，则需要添加新的状态和操作，并通过相应的操作来建立状态之间的联系。比如，我们可以双击左侧状态列表的最后一个状态 Postponed 的下一个空行，输入“退回测试部”，然后回车。你会发现在上方和左侧的灰色区域都会出现一个新的状态：退回测试部。如下图所示

From / To	Submitted	Assigned	Opened	Resolved	Closed	Duplicate	Postponed	退回测试部
Submitted								
Assigned	Assign						Assign	
Opened		Open		Reject	Re_open			
Resolved			Resolve					
Closed	Close	Close		Validate			Close	
Duplicate	Duplicate		Duplicate	Duplicate	Duplicate			
Postponed	Postpone	Postpone	Postpone					
退回测试部								

我们把这个新增的状态定义为：当项目经理分配缺陷时，或者当开发人员准备解决缺陷时，如果发现缺陷描述的不够准确，或者提交的缺陷记录本身存在其他的，可以将这条缺陷退回测试部门，要求测试部门重新描述或进行相应的修改。

如果在矩阵中来看，也就是说在 Submitted 之后，可以转到退回测试部状态，或者在 Assigned 之后，也可以转到退回测试部状态。保存一下这些变化，我们还需要添加一个新的操作来完成这些状态之间的关联。

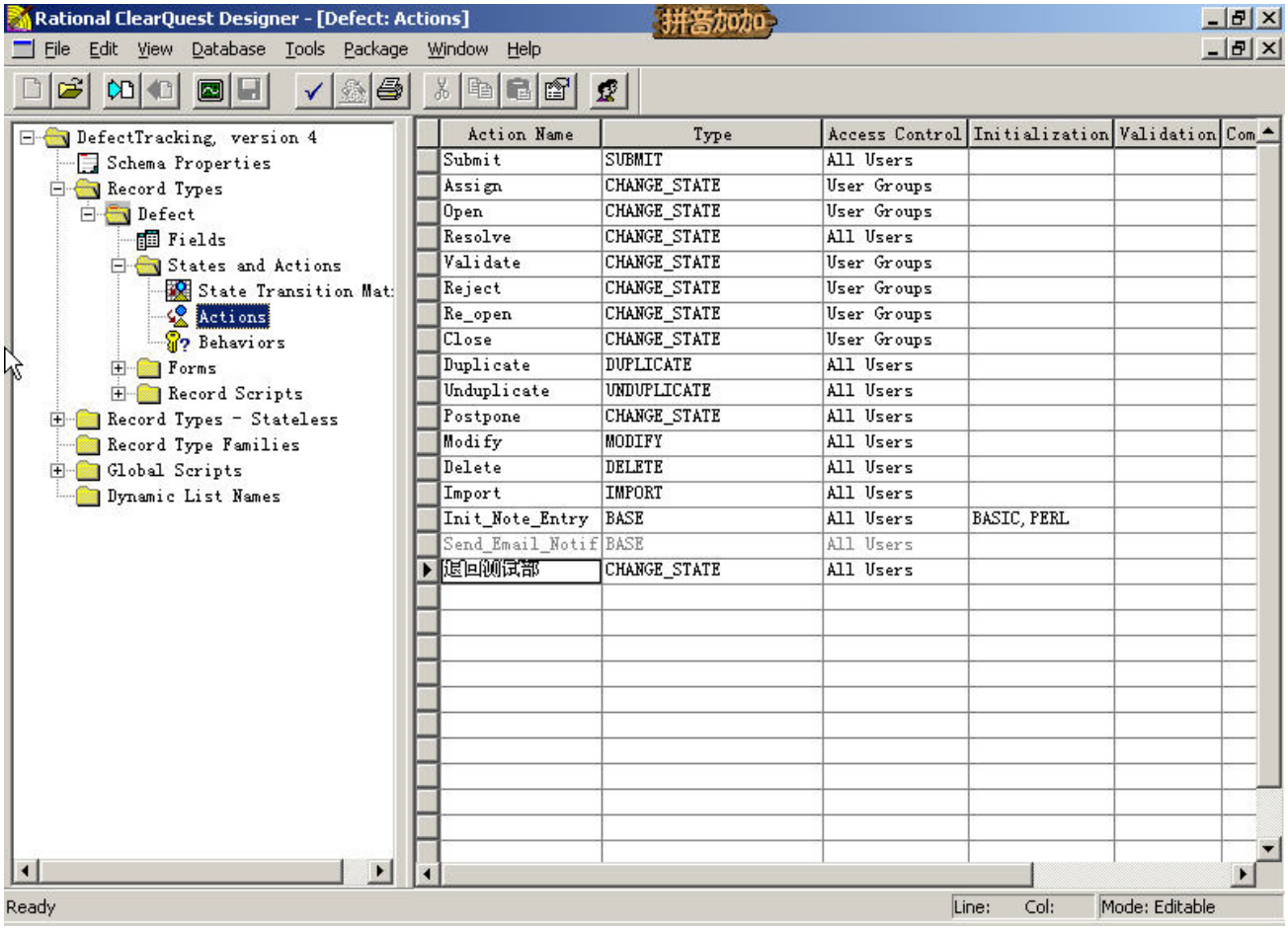
那我们继续进入 Action 的维护界面，如下图所示，点击红色框内的“Actions”

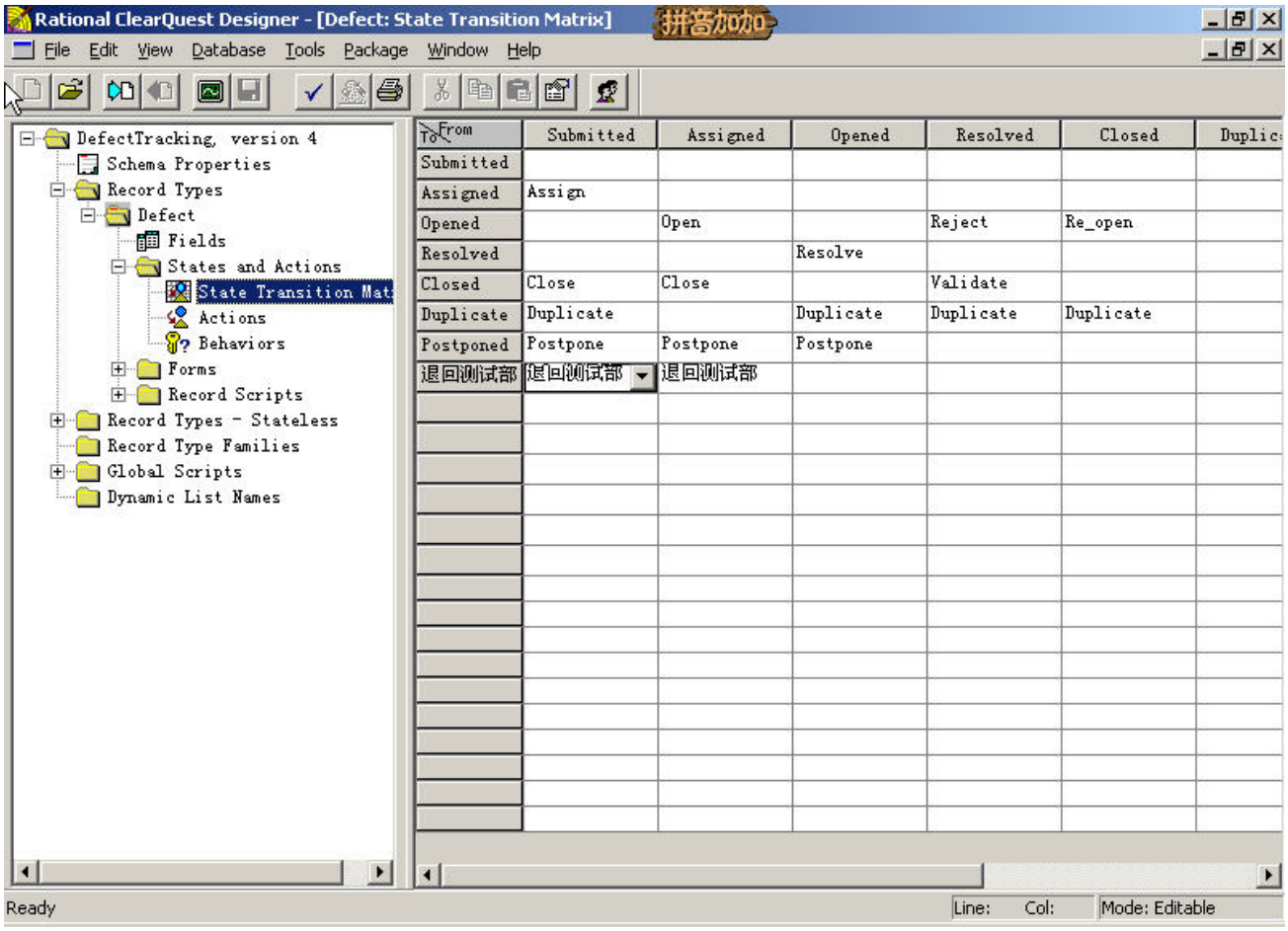


在这张图的右侧矩阵中，显示了 CQ 中现有的 Action 的列表。如果我们需要添加一个新的 Action，则可以在最后一条记录下面的空行出点击鼠标右键，选择“Add Action”。然后在弹出的 Action 界面中输入 Action Name——“退回测试部”，如下图所示

然后再点击“State”页面，选择相应的“源状态”和“目标状态”，如下图所示

这里我们根据前面的定义，选择“Source（源状态）”为 Submitted 和 Assigned，“Destination State（目标状态）”为“退回测试部”。关闭当前界面。这时，你可以看到在 Actions 矩阵中出现了新的 Action，而在“State Transition Matrix”中也可以看到相应的状态有了进行关联的操作，如下图所示





另外，刚刚在 Actions 矩阵中，我们还看到了有一个名为“Access Control”的字段，这个字段中保存的信息，就是用来控制不同组的用户对于操作的权限的。比如我们刚刚新增的“退回测试部”操作，只有开发部门的项目经理和程序员可以执行，而测试人员是没有这个权限的，我们就可以利用这个功能来进行相应的设置，如下图所示

Action Name	Type	Access Control	Initialization	Validation	Com
Re_open	CHANGE_STATE	User Groups			
Assign	CHANGE_STATE	User Groups			
Open	CHANGE_STATE	User Groups			
Close	CHANGE_STATE	User Groups			
Validate	CHANGE_STATE	User Groups			
Reject	CHANGE_STATE	User Groups			
Submit	SUBMIT	All Users			
Resolve	CHANGE_STATE	All Users			
Duplicate	DUPLICATE	All Users			
Unduplicate	UNDUPLICATE	All Users			
Postpone	CHANGE_STATE	All Users			
Modify	MODIFY	All Users			
Delete	DELETE	All Users			
Import	IMPORT	All Users			
Init_Note_Entry	BASE	All Users	BASIC, PERL		
Send_Email_Notif	BASE	All Users			
退回测试部	CHANGE_STATE	All Users			

SCRIPTS
ALL USERS
USER GROUPS

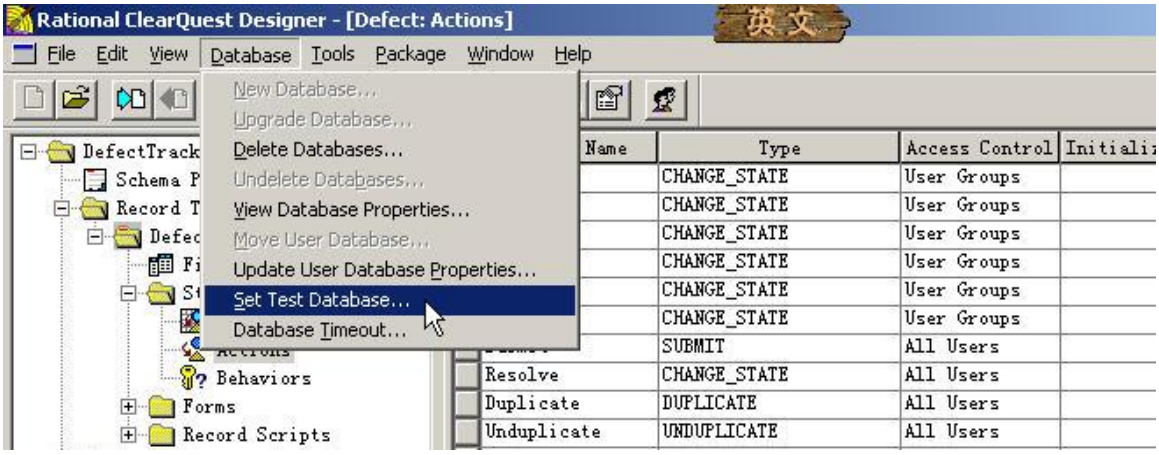
点击我们需要设置的 Action 的 Access Control 字段，然后选择“USER GROUPS”项，在弹出的用户组列表中选择“DevelopmentGroup”这个组，点击 OK 退出。这样，我们就赋予了 DevelopmentGroup 组的用户执行“退回测试部”操作的权限。

使用工具栏中的保存功能保存刚刚所作的变化。

8. 使用 Test Database 对变动进行调试

我们完成了缺陷管理流程的调整或变更了 Action 的权限之后，如何来验证这些变化的有效性呢？CQ 为我们提供了 Test Work 的功能，让我们可以在变化进入工作之前“先睹为快”。

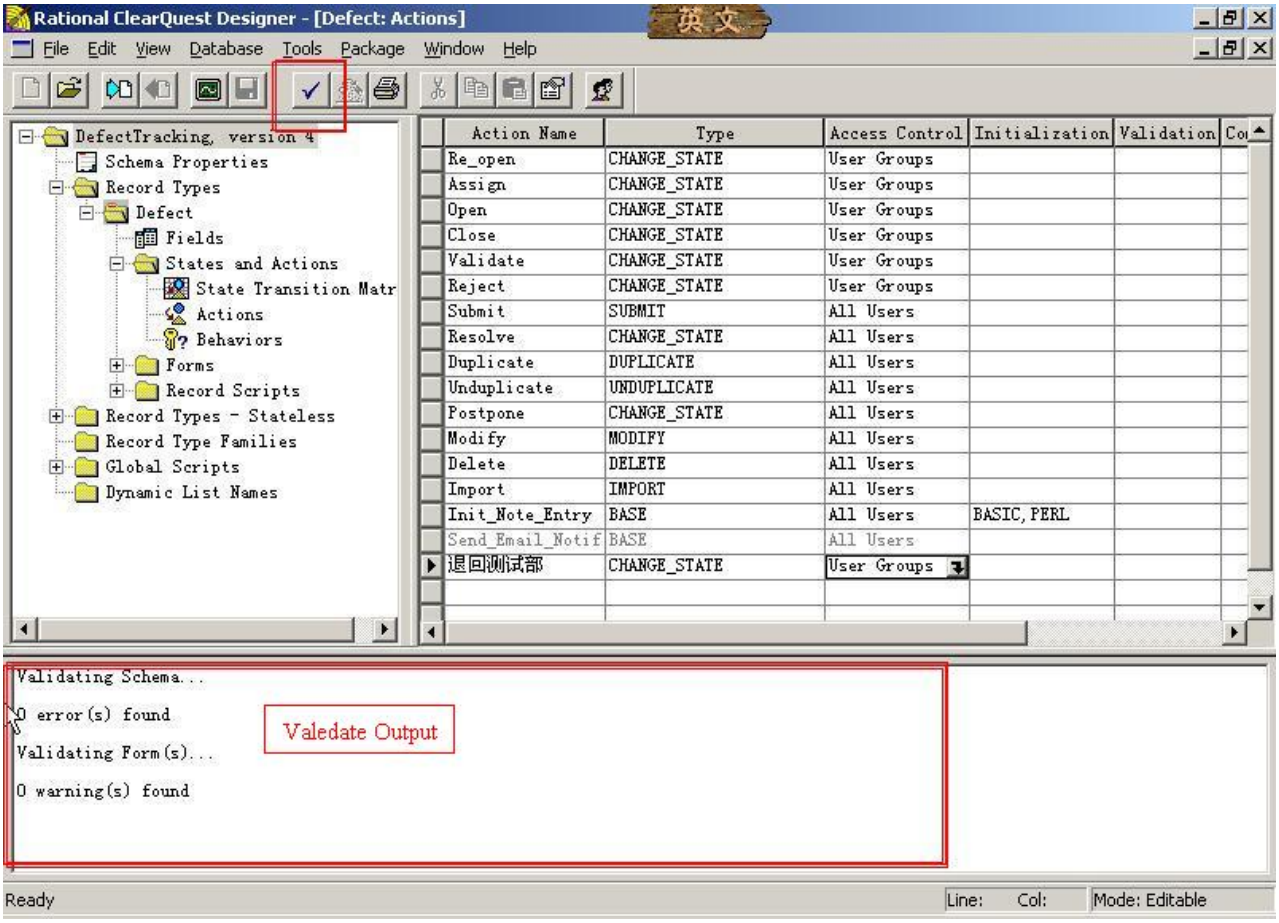
首先，我们要设置一个用来试验的数据库，如下图所示



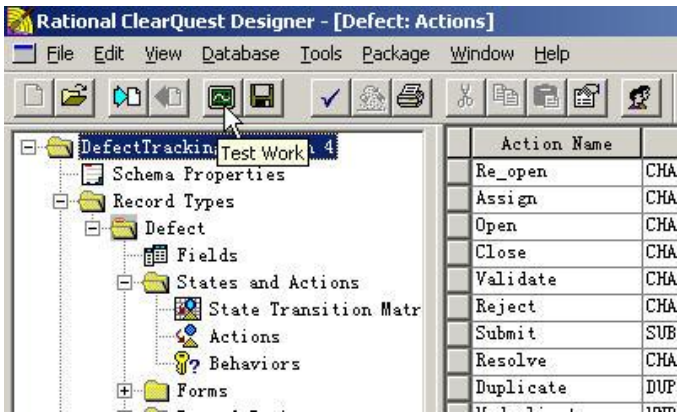
选择 Database 菜单下的 Set Test Database 菜单项，在弹出的 Test Database 窗口中，选择一个 Test Database，然后选择登录该数据库所使用的用户，在用户选择后，密码会自动填入。注意，这里我们选择的是 test 这个数据库，如果你仔细看过文档的前面部分，应该还有印象，这是我们专门建立的一个用来试验 CQ 变化的数

据库。

选择 OK 完成设置之后，点击 CQ Designer 工具栏中的 Validate 按钮，如果在屏幕下方的 Validate Output 中没有出现错误提示信息，则表明这次变化没有引入缺陷，可以进行测试了（当然，如果出现了错误提示信息，就要根据信息的内容来进行相应的调整）。如下图所示



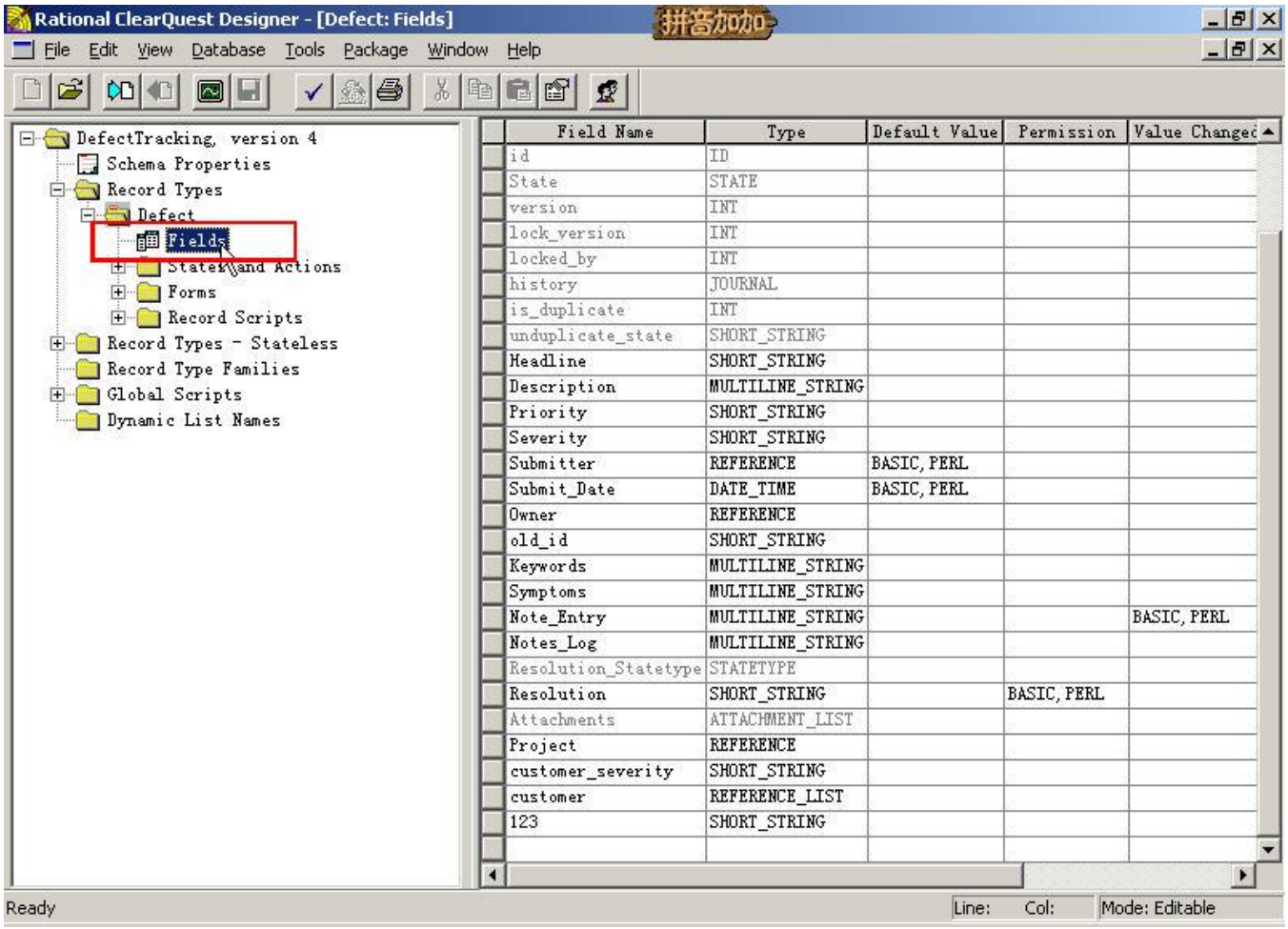
然后点击工具栏中的 Test Work 按钮，提交一个缺陷实际操作一下，看看是不是和你预想的一样。



9. 内容定制

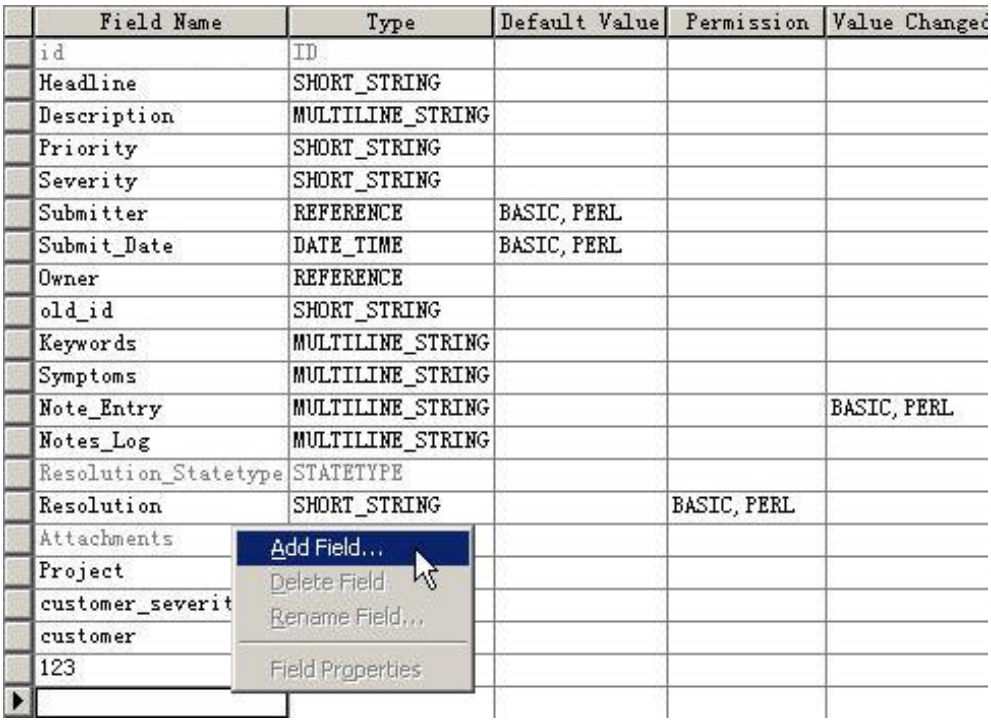
CQ 的提供的内容定制包括两方面内容，一是对数据库结构的调整，二是对用户界面的调整。例如，我们需要在提交缺陷的时候添加一个“缺陷来源”的项目，并希望提供不同的来源可供选择，比如需求阶段、设计阶段、编码阶段、实施阶段等。那么就需要先在数据库中增加一个相应的字段，并设置相应的默认可选值，再在用户界面中添加同这个字段相关的控件来供我们操作。下面我们就实际操作来看一下如何实现上面例子中的功能。

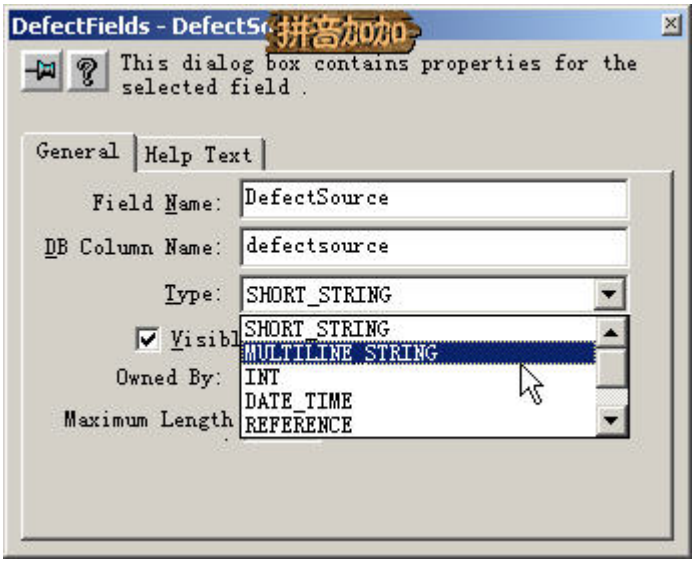
首先，我们进入 CQ Designer 的“Fields”项，如下图所示，在界面的右侧所列出的就是当前数据库中已经预设的一些字段。



在这个列表中，我们可以看到很多熟悉的身影，例如 Headline 字段是提交缺陷是的主题项，Description 是详细描述，Priority 是优先级，Severity 是严重程度……我们现在需要新增一个叫 DefectSource（缺陷来源）的字段。首先在列表的最后一条记录的下一行点击鼠标右键，选择“Add Field…”项，然后在弹出的界面中填写“Field Name”的值为“DefectSource”，“DB Column Name”由系统自动填写，由于我们

添加的这个字段要提供一个可供选择的列表，所以在“Type”项要选择“SHORT_STRING”（多行字符串）。退出当前界面。

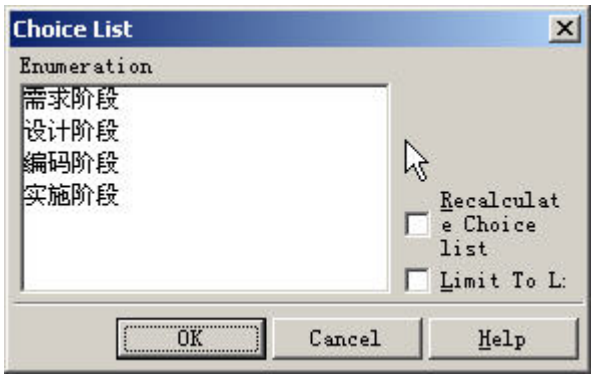




然后，我们需要在 DefectSource 字段中添加需要提供给用户选择的列表项。点击 DefectSourse 字段最右侧的“Choice List”项的箭头按钮  ，选择“Constant List”项，然后在弹出的界面中添加需要使用的列表项，如下图所示

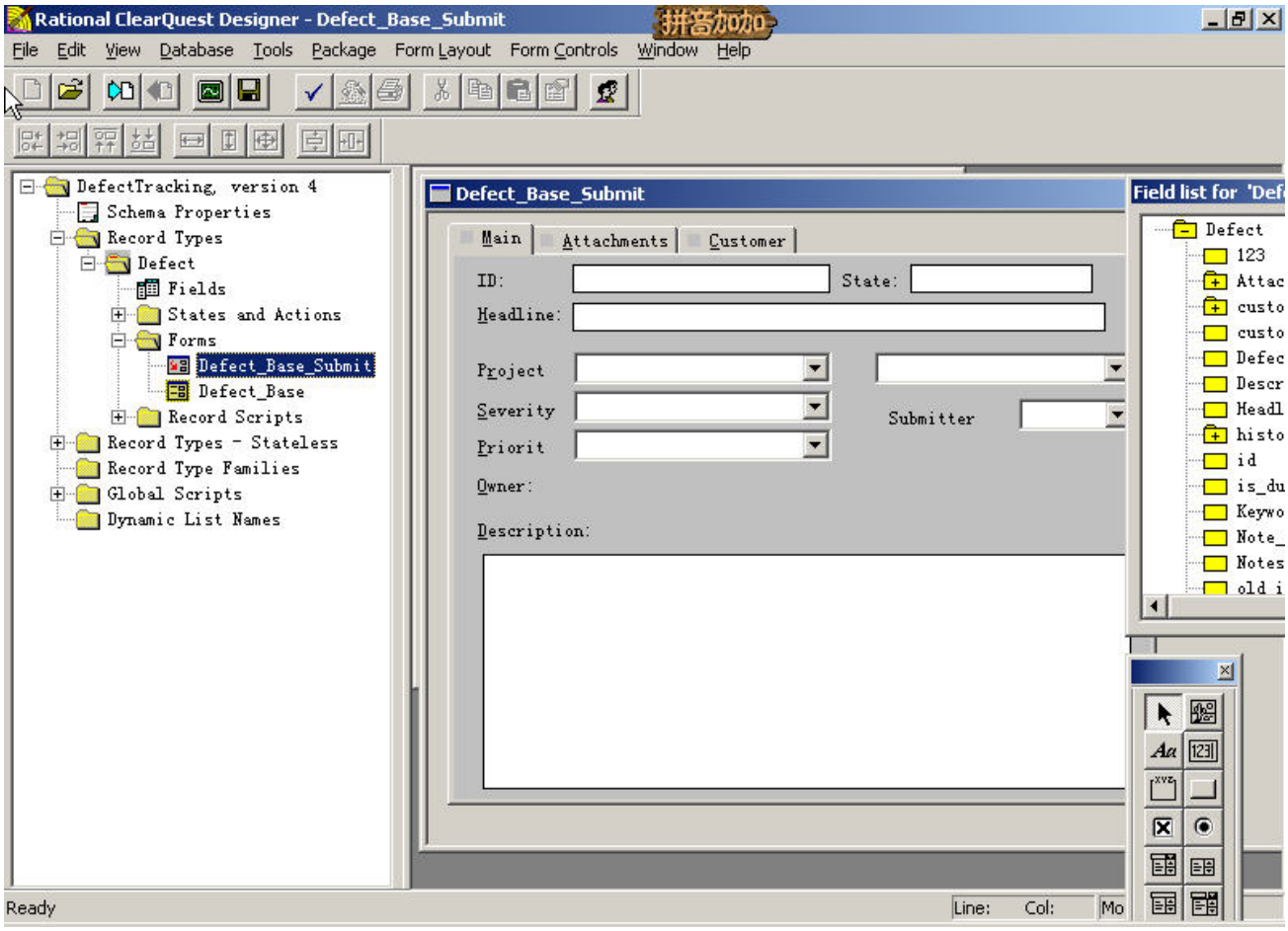
Field Name	Type	Default Value	Permission	Value Changed	Validation	Choice List
id	ID					
Headline	SHORT_STRING					
Description	MULTILINE_STRING					
Priority	SHORT_STRING					CONSTANT_LIST
Severity	SHORT_STRING					CONSTANT_LIST
Submitter	REFERENCE	BASIC, PERL				DEFAULT
Submit_Date	DATE_TIME	BASIC, PERL				
Owner	REFERENCE					DEFAULT
old_id	SHORT_STRING					
Keywords	MULTILINE_STRING					CONSTANT_LIST
Symptoms	MULTILINE_STRING					CONSTANT_LIST
Note_Entry	MULTILINE_STRING			BASIC, PERL		
Notes_Log	MULTILINE_STRING					
Resolution_Statetype	STATETYPE					
Resolution	SHORT_STRING		BASIC, PERL			CONSTANT_LIST
Attachments	ATTACHMENT_LIST					N/A
Project	REFERENCE					DEFAULT
customer_severity	SHORT_STRING					CONSTANT_LIST
customer	REFERENCE_LIST					DEFAULT
123	SHORT_STRING					
DefectSource	SHORT_STRING					

SCRIPTS
CONSTANT LIST
DYNAMIC LIST
NONE



如果我们希望在提交和处理缺陷的时候可以使用这个字段，那么还需要在 CQ 提供给我们的操作界面中添加这个字段。

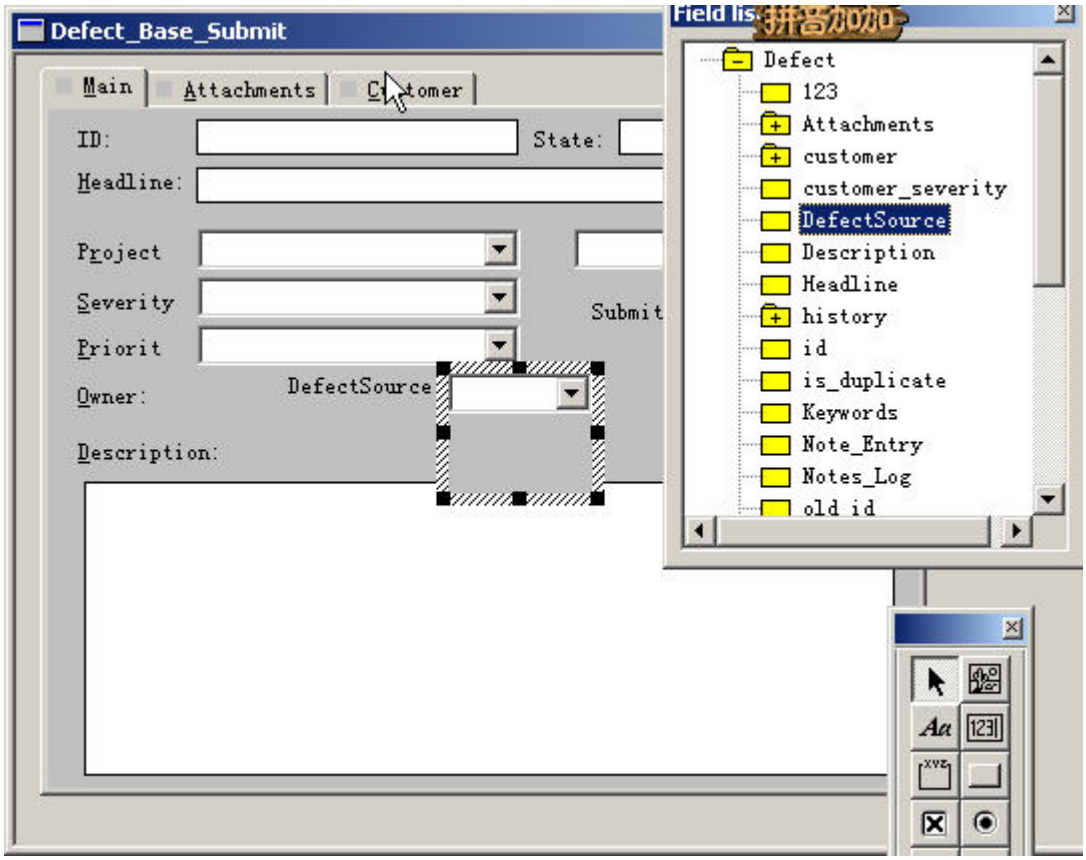
对于用户界面的调整，需要对“Forms”项中的 Defect_Base_Submit 和 Defect_Base 进行操作。其中 Defect_Base_Submit 是提交缺陷时使用的用户界面，而 Defect_Bas 是提交后对缺陷进行处理时使用的用户界面。我们以 Defect_Base_Submit 为例看一下如何实现前面讲到的例子，对 Defect_Base 的操作是相同的，我们就不多赘述了。如下图所示

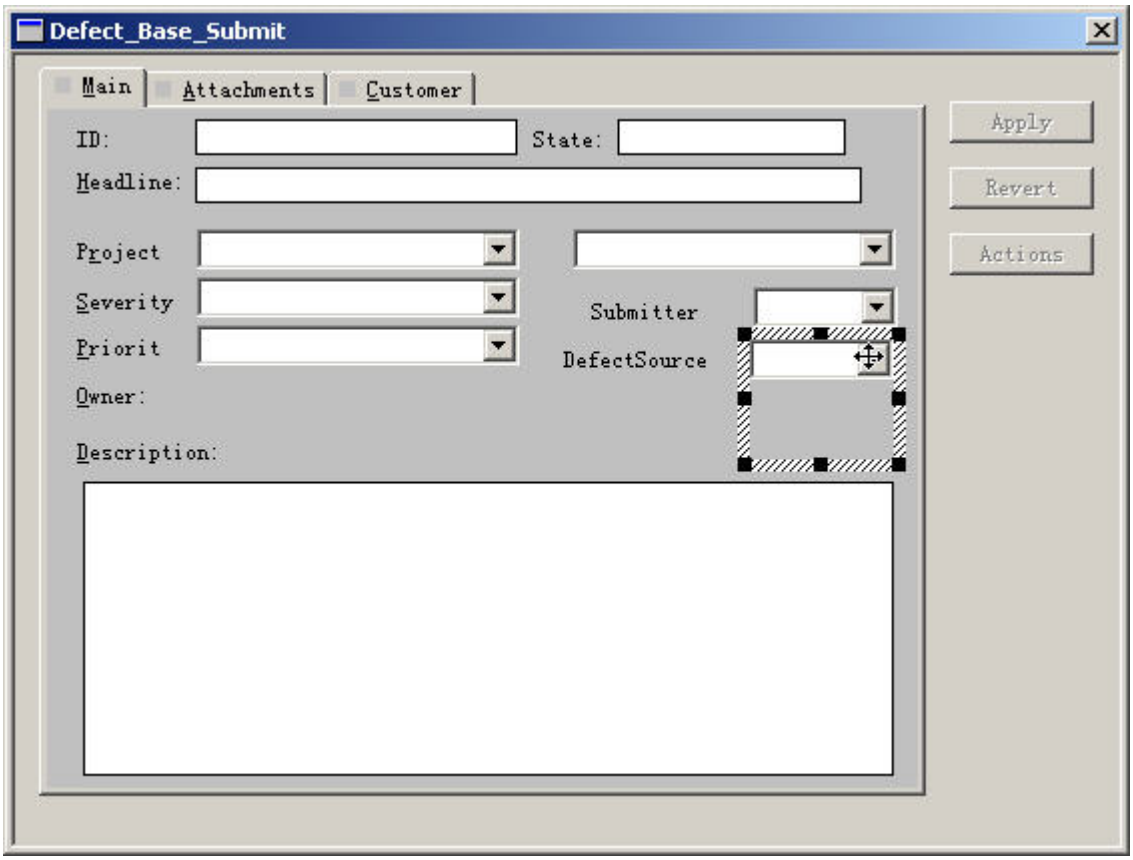


双击“Defect_Base_Submit”后，在屏幕的右侧可以看到三个面板，分别是中间最大的 UI 设计器，右上方的字段列表和右下方的控件面板。

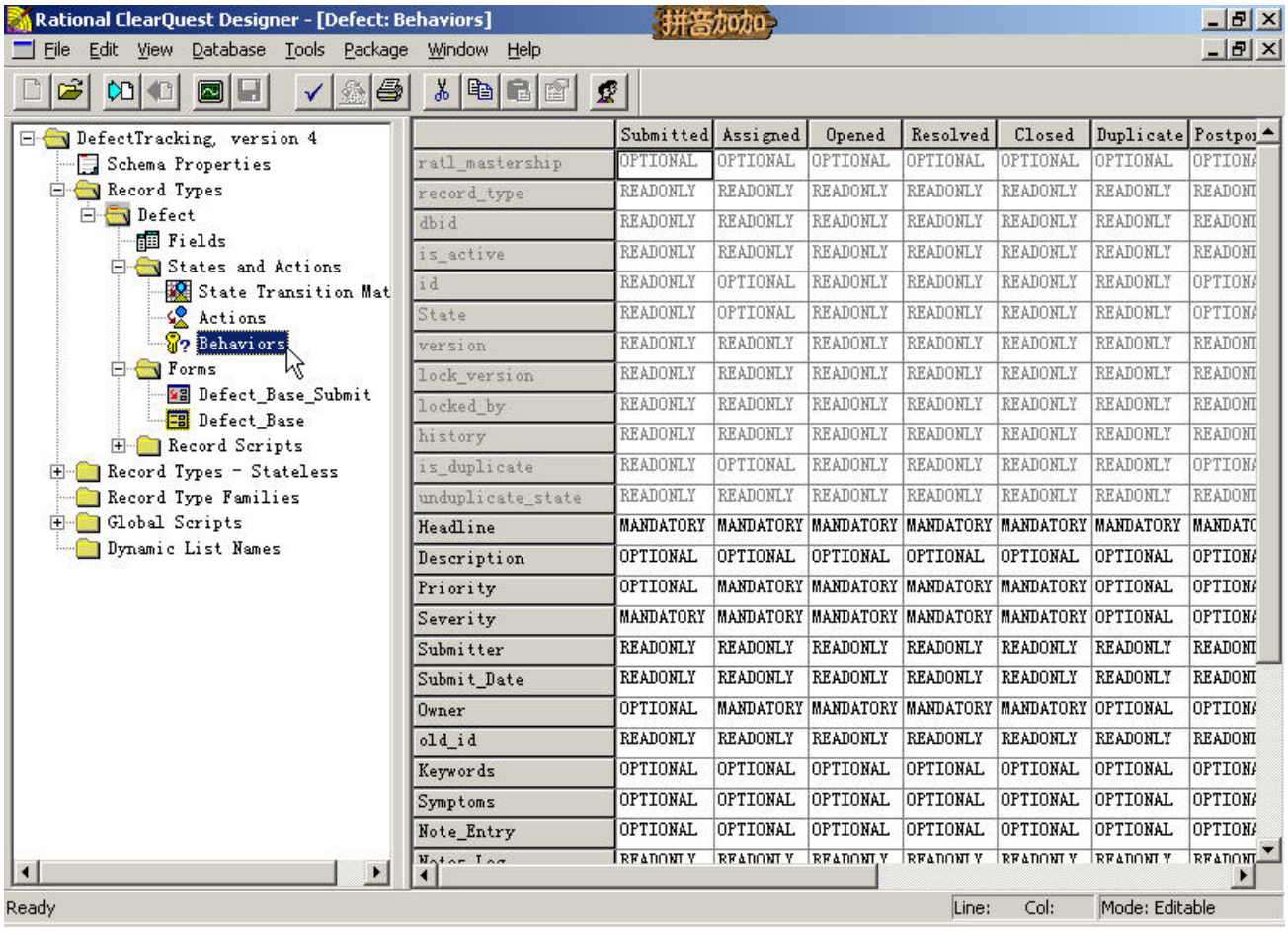
添加我们刚刚新增的 DefectSource 字段到 UI 中，可以使用两种方法实现，一是直接通过字段列表选择 DefectSource 字段，然后把它拖到 UI 设计器中，系统会根据字段的 Type 值来自动选择并生成合适的控件；或者也可以先在控件面板中选择需要的控件类型，然后再将控件关联到某个数据库字段上。对于第二种方法，因为需要开发方面的专业基础知识，所以我们更建议使用第一种方法来进行操作。我们下面就以第一种方法来演示一下这个操作过程。

我们首先选中字段列表中的 DefectSource 字段，然后使用鼠标将它拖至 UI 设计器中的空白区域，放开鼠标后，就可以看到在 UI 设计器中生成了一个新的控件，并且使用字段名的内容生成了相应的标签。之后我们再通过鼠标的拖放，将控件调整至合适的位置，就可以保存了。如下图所示





如果我们想进一步设置 DefectSource 字段为必填项，在提交缺陷的时候必须确定这个缺陷的来源，那么就需要通过“Behaviors”进行设置了。如下图所示



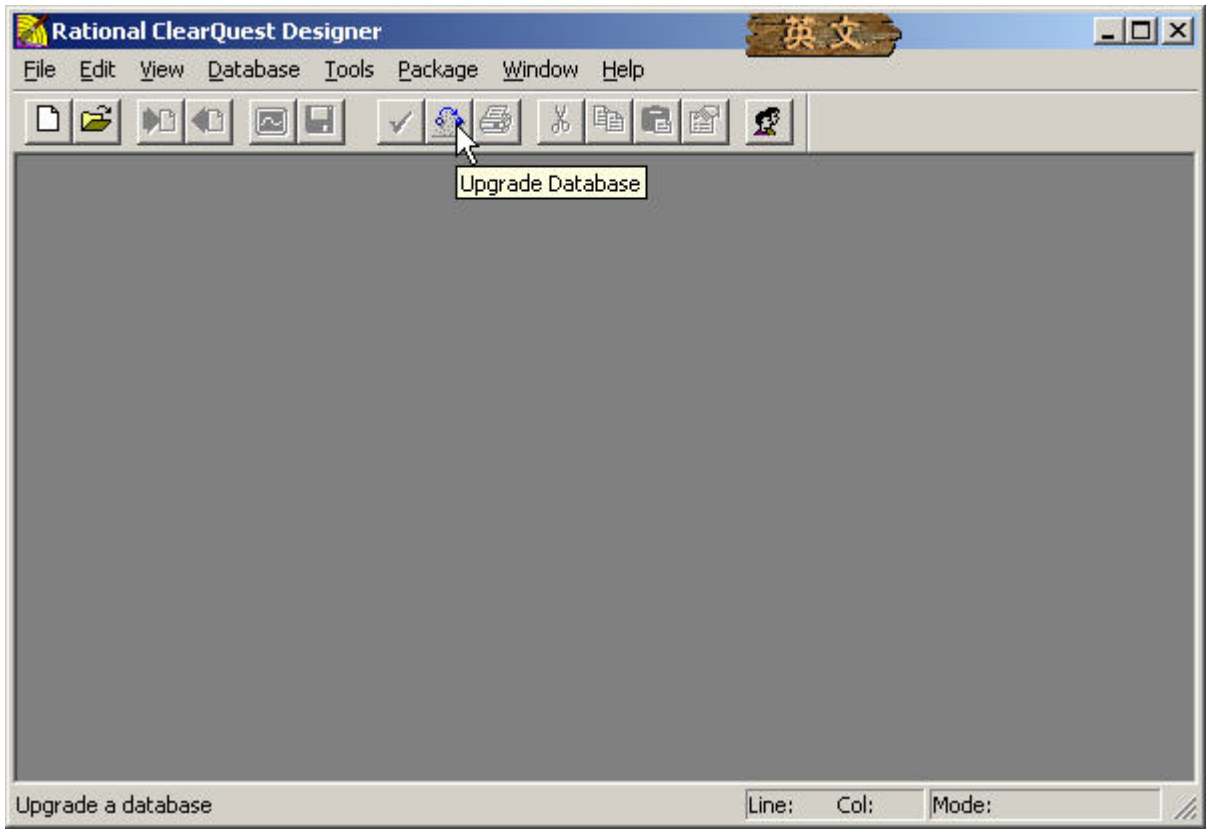
我们双击左侧 WorkSpace 中的中的“Behaviors”项后，在屏幕的右侧会出现一个矩阵，矩阵的上方是系统中现有的所有状态，而左侧则是系统中现有的所有字段，之间的内容是某个字段在不同的状态所应当遵守的规则。例如，对于我们新增的 DefectSource 字段，从提交缺陷开始，就必须要选择某项内容，在缺陷处理的整个过程中，这一项是不允许修改并且不允许为空的。所以，DefectSource 一个需要强制填写的项目，我们可以通过进行设置来添加这个规则。如下图所示

	Submitted	Assigned	Opened	Resolved	Closed	Duplicate	Postponed	退回测试部
Headline	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY
Description	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Priority	OPTIONAL	MANDATORY	MANDATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
Severity	MANDATORY	MANDATORY	MANDATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
Submitter	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Submit_Date	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Owner	OPTIONAL	MANDATORY	MANDATORY	MANDATORY	MANDATORY	OPTIONAL	OPTIONAL	MANDATORY
old_id	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Keywords	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Symptoms	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Note_Entry	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Notes_Log	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY	READONLY
Resolution_Statetype	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Resolution	USE_HOOK	USE_HOOK	USE_HOOK	USE_HOOK	USE_HOOK	USE_HOOK	USE_HOOK	USE_HOOK
Attachments	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Project	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
customer_severity	OPTIO		PTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
customer	OPTIO		PTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
123	OPTIO		PTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
DefectSource	OPTIO		PTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL

我们首先选择 DefectSource 同第一个状态——Submitted——的交汇点，也就是图中红框所示的位置，点击鼠标右键，选择“Mandatory”项，表示强制填写；然后在 DefectSource 同其他字段的交汇点如法炮制，选择“Read Only”项，表示只读——也就是说这个项目在提交后，就不允许修改了。

注意：如果在修改界面的时候选了必填的字段加入到界面中，然后删除，在运行的时候会提示有字段没有填，但是我们看不到这个字段，因为我们刚才删了，解决的办法是，在上面的界面中把我们刚才删的那个字段改为不是必填的就可以了

完成了上面的工作，保存并 Validate 后，我们就可以使用前面所说的 Test Work 功能来进行调试了。如果确认变更没有问题，则可以将 Test Database 中的数据同步到 Producation Database 中，如下图所示



首先选择工具栏中的“Upgrade Database”工具，然后会弹出一个提示信息，告诉你这个操作是不可逆的，一定要想好。我们确定之后就可以看到数据库列表了，

因为我们每次操作后都是将变化更新到 Test Database 中，而一直没有 Upgrade（升级）到 Producation Database 中，所以我们在实际工作中还是体验不到新的变化所带来的便利。我们现在选择第一个数据库（对于列表中各字段的含义，请参照前面已经讲过的内容），然后点击“下一步”按钮。

现在我们看到的是可供选择升级的版本的列表。我们在 CQ Designer 中进行流程或字段以及 UI 定制时，每次从 Test Database 中 Check Out 上次修改的结果到本地进行操作，之后再 Check In 回 Test Database 中，以保存这次的变化。这样就会产生一个新的版本，我们现在看到的就是 Test Database 中已经有的，比 Producation Database 中的版本新的版本。这时我们选择某个版本，点击“完成”按钮，如果可以看到下面的提示，那说明操作已经成功了。

附录：

CQ 中对于用户权限管理的原理

在上面的介绍中，我们看到了有四处地方同 CQ 的用户权限控制有关，分别是：

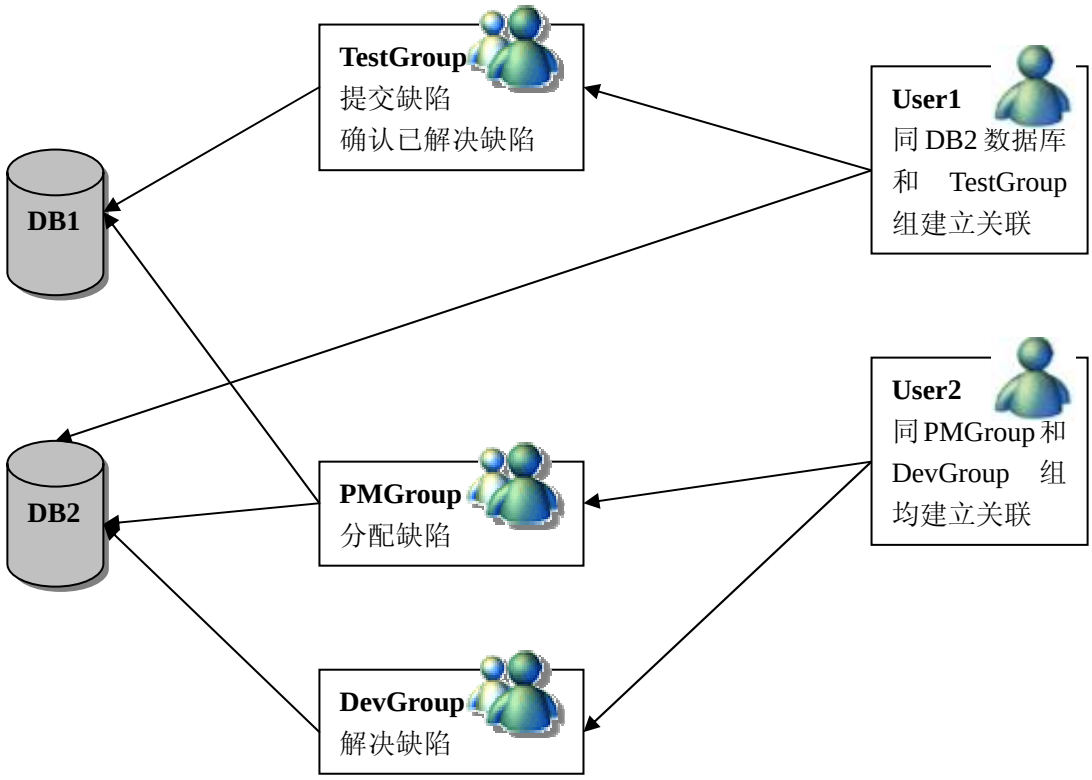
01. 维护用户信息时可以设置用户同数据库的关联；
02. 维护用户信息时可以设置用户同组的关联；
03. 维护组信息时可以设置组同数据库的关联；
04. 在 Action 列表中，可以通过 Action Control 来设置组同操作的关联。

这四处地方的设置，对于 CQ 用户的权限控制起着不同的作用。例如，本文上面所讲的操作都被保存在某个数据库中，我们将同缺陷跟踪管理流程有关的所有 Action 都设置了同某个组的关联——甚至包括提交缺陷的操作；每个组也同该数据库进行了关联；最后，新建一个用户，只将该用户同数据库关联，却不同组关联。这时，假如我们使用该用户从 CQ 客户端登录到数据库中，将只能建立查询、浏览数据，但却没有任何其它操作的权限；但是当我们将用户同组关联起来以后，再次登录到 CQ 中，就可以行使所关联的组在该数据库中具有的权限了。

通过上面的试验，我们就弄清楚了 CQ 中分级权限管理的原理：

01. 同用户关联的数据库用来控制用户能否通过 CQ 客户端或 Web 方式访问哪个数据库中的数据；
02. 假如用户关联了某个组或多个组，则可以登录到同这些组相关联的数据库中；

03. 假如每个 Action 都同相应的组进行了关联，则用户在登录到数据库以后，会根据所登录的数据库查找处所有有关联的组，再检查用户同这些组中的哪些组有关联，最后再根据这些同用户相关联的组所关联的 Action 来确定用户在自己当前登录的这个数据库中有哪些操作权限。如下图



在这个略显烦乱的示意图中，我们可以看出，当 User1 登录到数据库 DB1 的时候，由于他是 TestGroup 组的成员，所以可以执行提交缺陷和确认已解决缺陷的操作；而当他访问 DB2 时，可以浏览已有的缺陷记录，但不能对缺陷记录进行任何操作。当 User2 登录到 DB1 时，因为他是 PMGroup 的成员，所以能够执行为缺陷指定缺陷负责人的操作；而当他登录到 DB2 时，因为同时是 PMGroup 和 DevGroup 的成员，所以可以对 DB2 中的缺陷记录执行指定缺陷负责人和解决缺陷的操作。