Revision	Description
4/6/2010	Original

SQL-Hero Unit and Performance Testing

Introduction

Why do we need unit testing in the SQL world? The short answer is "for all the same reasons unit testing is beneficial in the non-SQL world." These benefits include but are not limited to: better adaptation to change (catch issues introduced by change early), detection of the unexpected (e.g. someone adds or drops some indexes and thus changes performance characteristics for some object), and catching errors much earlier in the process – errors that might normally be found at run-time instead. These are just a few examples we've observed where SQL-Hero has been brought in and adapted to solve specific needs.

How does SQL-Hero's support for unit and performance testing differ from some competitors? Here's a summary of what you can do with SQL-Hero's implementation:

- Run tests on a scheduled basis or on demand, and have a complete history of test results
- Generate notifications from the test results notifications that can be delivered by a number of means including e-mail
- Attempt to automatically generate tests against SQL objects that are missing them, using a shared testing policy; optionally try to use previously captured trace information to build tests. This is an important feature since it's not surprising that many developers are time-pressed and allocating dedicated time for building unit tests can be a challenge. Luckily SQL-Hero is very configurable in this department, and the global testing policy configuration file helps define rules for building tests.
- Opt out of testing certain objects, as needed
- Exercise tests when you commit object changes using the SQL-Hero editor tool, finding some kinds of problems immediately
- Define optional performance targets by object
- Compare test results against different types of "expected results": row counts, data checksum, etc. The most basic test is simply to assert that the object does not produce an error when run. Since this can be a common problem during the application development life-cycle, the concept of automatically constructing tests gains traction: so long as we ensure most of the object's contents "run," the results are secondary in many ways.
- View different kinds of reports that relate to testing, such as code coverage reports

And of course, the integration of the various tools is another benefit that's hard to ignore. This paper will cover examples such as being able to set expected test results right from the result set produced by a procedure.

A prerequisite for using unit testing with SQL-Hero is that you have access to an installed instance of the SQL-Hero server components. This is because both test meta-data and test results are stored in the repository. More details on installing SQL-Hero server components can be found in the whitepaper "Installing SQL-Hero."

Defining Tests

What kinds of objects support testing using SQL-Hero? Stored procedures can be "exercised" by calling them in a transaction that will be rolled back. Tables and views can be "selected from" to ensure they're functional (for a table, this can be useful if it uses computed columns, for example). A future release of SQL-Hero will support testing UDF's and also test via SQL scripts. In all cases, it's important to test objects that are "safe," meaning they can be wrapped in a transaction and have the transaction rolled back either inside or outside of the object, without producing an unexpected data change.

Right away this requirement may disqualify some kinds of objects. For example a procedure that does a complex data load using a cursor loop which performs BEGIN TRAN / COMMIT or ROLLBACK TRAN on its own, may not be appropriate for unit testing.

To disable unit testing on an object, one simply needs to flag the object by setting the "DoNotUnitTestCover" user-defined property to True (checked):



UDP's for RD-Development:up_SomeChangeProc									
🚽 Save 🕥 Cancel 🎦 Create UDP 😰 Script As Xml 🐚 Script To Clip 🔇 Previous 🌍 Next									
Category A									
Name	Value	Null	Global	ExtPro					
Demo08_Specific	Demo08_Specific								
IsForTaskTracker									
E Category : Built-in Testing Support (5 items))								
DoNotUnitTestCover			•						
TestRunExpectedResultChecksum									
TestRunParameters									
TestRunScheduledTimeout									

This UDP is "global" and as such applies in all databases, by object name. Another way you can set this property is from the "Test Parameter Values" screen:



📲 Parameters / Variables								
	OK 🕤 Cancel 🏘 Randomize 🞒 Sc	ript 🕂 UDP XM	L Test Yes	 Performance Target (ms) 	C			
	Variable	Туре	Size	Value	Null			
▶	@StringParameter	varchar	(100)	01U8QLXVAINR2FZPW1DNW5				
	@IntegerParameter	int			•			
	@BooleanParameter	bit		1				
	@CustomerTypeID	int		1				

Notice this screen is already showing some parameter values, related to testing. Did someone provide these? No: when the procedure was created by executing it in the SQL-Hero editor, test parameters were arrived at automatically.

One can control this behavior of trying to automatically create tests using the Manage -> Global Settings screen, available from the Editor toolbar menu:

🖞 SQL-Hero										
Editor Data Compare Schema Co	ompare Differences	History Noti	fications Create Data							
Category: RD Database: RD-Development Filter Filter Execute										
File▼ Edit▼ Tabs▼ Actions▼ Obj	ect ▼ Favorites ▼ Temp	lates • Manage •	👫 Find: count							
i 🖬 🖻 🍱 🖀 🗩 🔊 🤋	o 🛧 🔄 🗈 🗙	🙀 🚽 😫 🛛 Mar	nage Users							
Stored Procedures	OL (4) X	🚊 Ton	nbstone Settings							
🖃 🐏 up_SomeChangeProc		🗕 👌 Sou	rce Control Settings							
		🥑 Glo	bal Settings							
		🕀 Mar	nage UDP's							
Bependenter		📶 Ser	ver Logs							
Tables		👰 Rec	orded SQL History							
🕎 Global Settings			×							
Stored Procedure Test Requirements										
Must have test UDP and attempt to au	o-generate	•	# random tries							
Maximum wait (milliseconds)	8000									

Options here let you opt out entirely from auto-generation of tests, force developers to provide test parameters manually, or attempt to auto-create tests. The default number of random combinations tried is "12" unless specified. The maximum wait is for *all* test creation attempts and prevents you from having to wait more than a fixed timeframe when your object changes are committed in the SQL-Hero editor.

Also, if you follow the standard that foreign keys either end in "ID" or "Key", SQL-Hero is smart enough to properly pick foreign key values that are known to exist. For example, in the procedure below:



If the table CustomerType exists, test values for the @CustomerTypeID parameter are determined based on what exists in the CustomerType table's CustomerTypeID column.

Finer grained control is also possible by modifying the TestingPolicies.xml file, located in your SQL-Hero application directory. If you're using SQL-Hero server components, the server that your client is referencing is used to pull the TestingPolicies.xml file, enabling you to make changes in one single location (i.e. the server copy) and have it apply, enterprise-wide.

The default TestingPolicies.xml file contains some commented out examples of how it can be used to tailor testing behavior.

```
<?xml version="1.0" encoding="utf-8" ?>
<ROOT>
           <!--
                       Insert your customized testing policy configuration here.
                      The version of this file maintained on your SOL-Hero applcation server is taken as the preferred source, else
your local copy
           <!-- A plug-in gives you maximum control over the test generation process - you can preview, set values based on your
own rules, etc. -
           <PLUGIN>
                      <assembly_file></assembly_file>
                      <TYPENAME></TYPENAME>
           </PLUGIN>
           -->
           <!-- Deals with specific objects (procedures)
           <OBJECT>
                      <NAME></NAME>
                       // OR - use a name regular expression to match against
                      <PATTERN></PATTERN></PATTERN>
                      // For the given name/pattern, can be excluded entirely from testing (will be marked as do not test)
                      <IS_EXCLUDED>True</IS_EXCLUDED>
           </OBJECT>
           <!-- Deals with specific parameter names
           <PARAMETER>
                      <NAME></NAME>
                       // OR - use a name regular expression to match against
                      <PATTERN></PATTERN>
                       // OR - match a specific UDP value
                      <UDP><NAME>UDPName</NAME><VALUE>True</VALUE></UDP>
                      // Can optionally associate only to a specific parent object as well
<PARENT_NAME></PARENT_NAME>
// OR - use a name regular expression to match parent object name against
<PARENT_NAME_PATTERN></PARENT_NAME_PATTERN>
                      // Can optionally set the maximum probability of using Null
<NULL_PERCENTAGE></NULL_PERCENTAGE>
                      // Use a constant value always
                      <VALUE></VALUE>
                      // OR - use a random value within a range
                      <VALUE_START></VALUE_START>
                      <VALUE_END></VALUE_END>
// OR - use a SQL query to get a range of values from which one is randomly picked
                      <SQL_LOOKUP></SQL_LOOKUP>
           </PARAMETER>
           <!-- Deals with all parameters of a particular data type (parameter name matching performed first)
           <DATATYPE>
                      <TYPENAME></TYPENAME>
                      // Can optionally set the maximum probability of using Null
<NULL_PERCENTAGE></NULL_PERCENTAGE></Pre>
                       // Can optionally associate only to a specific parent object as well
                      <PARENT NAME></PARENT NAME>
                       // OR - use a name regular expression to match parent object name against
                      rent_name_pattern>/parent_name_pattern>
                       // Use a constant value always
                      <VALUE></VALUE>
                      // OR - use a random value within a range
<VALUE_START></VALUE_START>
                      <VALUE END></VALUE END>
                      // OR - use a SQL query to get a range of values from which one is randomly picked
                      <SOL LOOKUP></SOL LOOKUP>
           </DATATYPE>
           -->
           <1--
                     a proc parameter that ends in areaid, populate using a select from area table that is filtered
           Example:
           <PARAMETER>
                     <PATTERN>AreaID$</PATTERN>
                      <SQL_LOOKUP>SELECT AreaID FROM Area a WHERE a.AreaAbbr LIKE 'A%'</SQL_LOOKUP>
           </PARAMETER>
                     a sample plug-in, put in the program files\sqlhero\plugins directory
           Example:
           <PLUGIN>
                      <ASSEMBLY_FILE>SampleTestingPlugIn.dll</ASSEMBLY_FILE>
                      <TYPENAME>SampleTestingPlugIn.MyTestPolicy</TYPENAME>
           </PLUGIN>
```



As you can see, you can control what objects will be tested, and even what values can be used for certain parameters.

You may be wondering what's the difference between the "TestRunParameters" and "NullRunParameters" user-defined properties. NullRunParameters is used mainly to support certain code generation tasks, but by having it, it too can serve as a unit test as well. NullRunParameters are generally intended to run a procedure and produce its expected result set (empty or not), without creating side-effects. This setting is global in the sense it will apply to all databases in which an object of that name appears. TestRunParameters on the other hand are database-specific, since the expected results may vary depending on the database.

Another way that you can set test parameters for an object is using the "Test Parameter Values" screen, as illustrated here:

👔 Parameters / Variables									
	nance Target (ms)	-	Test Yes	tript 🕘 UDP XML	🞊 Randomize 🗿 Sc	🛃 OK 🕤 Cancel			
Null	Value		Size	Туре	ariable	Va			
	VAINR2FZPW1DNW5	01	(100)	varchar	r	@StringParamete			
•				int	er	@IntegerParameter			
		1		bit	ter	@BooleanParame			
		1		int	D	@CustomerTypel			
ł		1		int	D	@CustomerTypel			

This screen lets you provide explicit values for the parameters, or attempt to arrive at values using a randomization strategy (Ref.). You can also script out the SQL that would run the current unit test (SQL value). You can also script out some XML that, if executed in the SQL-Hero editor, would set the testing user-defined properties (). From the example above, this would produce:



You could execute this comment in a different database, for example, to set the same TestRunParameters UDP in that database also.

The "Script" command (a) is useful for a number of reasons, one of which being that you can get the result set that the procedure would produce from the test and then define that as the "expected result" in terms of data, when the test is exercised. Doing this is rather easy: click the "Script" button, resulting in text such as this appearing:

	RD-Development:up_SomeChangeProc	×	RD-Development:SQL (9)* ×	RD-QA:SQL
BEC EXE IF	IN TRAN; SET ARITHABORT ON; CUTE [dbo].[up_SomeChangePr @@TRANCOUNT > 0 ROLLBACK TF	; roc] RAN;	'OHTGE14QWE2',154119276	,NULL,1;

Executing this script may result in a grid such as this:

Cı	ustomerTypeDesc	CustomerTypeCode		
1	Primary	PRI		

One of the context menu (right-click) options available on the result set is to set it as the expected result for the current test:

		2	Close All Close To File To Clipboard Grid	Ctrl+Shift+F9 Esc	• •
CustomerTypeDesc	Customer	\$	Set As Test Resu	t	
1 Primary	PRI	<i>a</i> ta	Find	Ctrl+F	
			Join Row To	Ctrl+J	
		D.	Preview Data	F11	
		0	Relabel Tab	Ctrl+Shift+T	

Note that it is possible to have more than one test specified for an object, but to do this, you cannot use the Parameters / Variables screen. Instead you need to modify the TestRunParameter UDP directly, as illustrated here:

B IIDP's for PD-Davelonmentum SomeChangeProc									
🛃 Save 🕙 Cancel 📋 Create UDP 횔 Script As Xml 斗 Script To Clip 🔇 Previous 🜍 Next									
Category 4									
Name	Value	Null	Global	ExtPro					
Demo08_Specific									
IsForTaskTracker									
- Category : Built-in Testing Support (5 items)								
DoNotUnitTestCover			•						
TestRunExpectedResultChecksum	175412327								
TestRunParameters	<root></root>								
TestRunScheduledTimeout	<tem>NULL,NULL,NULL,1</tem> <tem>NULL NULL 12</tem>								
TestRunTargetMilliseconds									
- Category : General (5 items)									
Definition	TODO								
		_	_	_					

Running Tests

There are a number of ways that tests become a part of different processes in SQL-Hero. For example, if a test is present on a stored procedure, it is exercised whenever you try to commit a change for it when using the SQL-Hero editor. If you've introduced a problem, for example, you can receive a message like this:



This is what we received when trying to commit the change shown below, in a database that does not yet have the Customer table:



Note the message shown at the bottom: this is what we received when we said "No" to the above dialog box when asked about setting up a test parameter manually.

Another place where tests are evaluated is when you use the "Test, Copy and Rollback" feature in the SQL-Hero editor:

SQL-Hero Whitepaper Series Codex Enterprises LLC Copyright © 2010



Picking this, you're prompted for a database in which to copy the object, run its unit test, and roll back the schema change which copied it. This can yield useful information such as seen here:

ObjectName		ObjType	Duration	TestType	PercentOfTarget	ErrMessage	ErrNumber	SQLParms	
1	[dbo].[up_SomeChangeProc]	Р	26	2	NULL	Invalid object name 'CustomerType'.	208	'0HTGE14QWE2',154119276,NULL,1	4/3/2

In this case we tried to copy up_SomeChangeProc to a database that did not have the CustomerType table. After this has been run, up_SomeChangeProc was not moved or altered in the target database.

Testing is built to leverage SQL-Hero's powerful template engine. As such, you can run tests directly against one or more objects, using the "Performance and Unit Test SQL" template, as seen here:

🕊 SQL-Hero										
Editor Da	ta Co	mpare Schema Compare	Differences	History	Notifications	Create Data	Testing	g Region Diffs	Tracing Monitoring	Settings
Category: RD		▼ Database: RD-De	velopment	•	🍸 Filter 🚰	Execute (F	5) Nar	me Filter:	 Content Search 	
File - Edit -	File - Edit - Tabs - Actions - Object - Favorites - Templates - Manage - 🏦 Find: count - Replace: - 🔽 🗟 🖞 🦻 🦞 🦉 🖬 🛔									
I										
Stored Procedures RD-QA:SQL (7)*- X RD-Development:up_SomeChangeProc X										
E- 👷 up_So	2	Refresh	Ctrl+R	FROM sy	s.objects	WHERE objec	t_id	= OBJECT_ID (N'[dbo].[up_SomeCh	angeProc]') AND
🕀 🦢 Pr	1	Create New	Ctrl+N	[up_Som	eChangePro	c]				
🕀 🦢 D	×	Delete	Delete							
- Tables		Move	Ctrl+Shift+E	edProce	dure [dbo]	.[up_SomeCh	angeP	Proc] Scri	pt Date: 04/03/201	.0 14:34:02 ****
		Copy	Ctrl+Shift+P							
	4 ¹ }	Run	Ctrl+F5							
	1	Compare	Ctrl+Shift+O	ON	ON					
	۳ ۲	Previous Object	Alt+F11							
	21	Locate Object (Pick Database)	Ctri+F2	up_Some	ChangeProc					
		Auvanceu	•	er varch	ar(100) .t					
		UDPs	Ctrl+F4	eter bi	t					
	2	Copy, Test and Dollhads		ID int						
	Sin .	Add to Build		******	********	********	*****	*****		
	<u></u>			geProc						
		Show History		Istrato	r					
	" 	show resung history		ustomer	related d	ata	_			
		Template for Selected	•	Scrip	ting, Builds and D	DL	•			
	D.	Check In		Ente	rprise Application	Development	•	CodexFramework	- Multiple Result Set Procedu	re Wrapper (C#)
	цэ <u>р</u>	Check In All		Infor	mational			CodeXFramework	- Search Screen (C#)	(2.12)
	-*	Undo Check Out		In-In	ie 1-SQL and DML		•	CodeXFramework	- Stored Procedure Class Wra	apper (C#)
		SELECT ct	.CustomerTy	peDesc,	ct.Custome	rTypeCode,	c. –	Standard UDBs for	unit rest SQL	
		FROM Cust	omerType ct					Stanuard ODPS 10	r stored procedures	

Selecting this prompts you with template parameters:

Template Parameters (RD-Development)							
Table Name	[
Test Types							
Select Results							
🔲 In Outer Transactio	on						
<u>О</u> К		<u>C</u> ancel					

Using the defaults (except for checking the "Select Results" checkbox) yields a SQL script that when run, actually invokes the test or tests. Running the script could result in output that looks like this:

	ObjectName	ObjType	Duration	TestType	PercentOfTarget	ErrMessage	ErrNumber	SQLParms	EventDate	RowNumber
1	[dbo].[up_SomeChangeProc]	Р	0	2	NULL	NULL	NULL	'0HTGE14QWE2',154119276,NULL,1	4/3/2010 21:49:03.010	1

In this example, we successfully executed the test and the test took less than 1 millisecond (seen in the "Duration" column).

Another common way that tests can be run is on a schedule. Testing schedules can be configured on the Testing tool:

👯 SQL-Hero				
Editor Data Compare	Schema Compare Differences Histor	y Notifications	Create Data Testing	Region Diffs Tra
Report: Recent Errors	✓ RD-Development	▼ Execute	Filter Schedules	M Apply Trace
Last Session Only	2/27/2010 🔽 to/_/	•		
i 📰 I4 4 0 of ▶) 4 🛛 🖻 🖨 🗐 🛄 🖓 - 10	0% -	Find Next	

In the example below, we're going to run all types of testing on the RD-Development database, starting next at midnight on 4/2/2010. It will run daily, and will include results on both successful completion and errors (if "Errors" was checked, only errors would be recorded). Optional name filters can be applied, and an optional filter can be used which causes only objects changed since a certain date to be included in the test. A global timeout can also be specified which is applied to every test case that does not have an overridden timeout period (this can be set using the "TestRunScheduledTimeout" user-defined property). In addition, this scheduled test will include a "creation phase" where objects that do not already have tests will go through the process of having tests automatically generated, where possible – here, at most 5 parameter combinations will be tried.

Init Testing / Performance Profiling Databases / Schedules										
🛃 OK 🕙 Cancel	🛿 OK 🕥 Cancel Apply 🔀 Delete 😰 Refresh 🖾 Kill SPID									
Database V	Next Run	Frequency	Туре	Errors	Include Name	Exclude Name	Changed Since	Test Timeout (ms)	Create Test Tries	Status
RD-Development	04/02/2010 12:00 AM	Daily /	All					30000	5	Scheduled

The "Kill SPID" button is only enabled when the scheduled test is actually running and has reached the stage where tests are being run. The object name currently being tested is shown:

Create Test Tries	Status	Current Object	Current SPID	
5	Scheduled			
5	Scheduled			

"Kill SPID" becomes useful if you believe the test is hung, taking too long, or otherwise causing a problem.

The results of scheduled testing are often effectively reported using SQL-Hero Notifications. Details on setting up notifications are covered in a different whitepaper. Another way to get at the results is using the reporting capabilities of the Testing tool, covered later.

Yet another way in which you can exercise unit tests is using the sqlheroproj project type, under Visual Studio (requires Developer edition or higher of SQL-Hero). In the example below, we're creating a new SQL-Hero project in Visual Studio 2008 that will be specifically for unit testing.

SQL-Hero Whitepaper Series Codex Enterprises LLC Copyright © 2010

New Project					? ×		
Project types:	Templates:	1.	IET Framework 3.5	•	0 0-0- 0-0- 0-0-		
Smart Device	Visual Studio installed tem	plates					
Office	BuildProject						
CodeX							
Reporting	My lemplates						
SSIS_ScriptComponent	Search Online Templates						
SSIS_ScriptTask							
WCF							
Workflow							
SQL-Hero Project							
BuildProject							
Other Project Types	1						
Test Projects							
	J				_		
J							
Name: RD.bBTests							
Location: C:\Users\Administra	tor\Documents\Visual Studio 2008\	Projects	•	Brows	e		
Solution: Create new Solution	•	Create directory	for solution				
Solution Name: RD.DBTests		Add to Source Co	ntrol				
			ОК	Cano	:el		

The next step is to open the designer for the BuildItem.sqlherogen file:

ංලි (RD.DBTe	sts - Mi	icrosoft \	/isual S	tudio (A	dministra	ator)										
Fi	e Edit	View	Project	Build	Debug	Tools	Test /	Analyze	SQL-Hero	Window	Help						
16		- 💕 🛛	3 🖉	አ 🖻	B 10) - (ii -	- @ -	■ ►	Debug	→ Any	(CPU		- 🖄	TestDoc	- 🗟 :	🚰 🗹	à 🛠 💽 🖸 - 🖕
	Build	item.so	lheroge	Start	t Page											• ×	Solution Explorer - Solution 'RD.DBTests' (1 pro
Serv	🖉 🕅 Preview 🐚 Add Template 🔮 Manage Data Sources 🔟 History																
er Exp	Project name: SQL-Hero application server:										Solution 'RD.DBTests' (1 project)						
lore	Description:										BuildItem.sqlherogen						
7			🗌 Log g	enerate	d content			Append t	o log		Halt on e	error	🗌 Only I	.og Errors / Warnings			
Toolbo	Enable	d S	equence	Des	cription	Template	Data S	iource I	nput Control	Source C	Control	Output Contro	ol Parallel				

Next, fill in the fields as appropriate. Here we're using "localhost" which assumes we're running this from the same box on which the SQL-Hero server components are installed. After that, click on "Manage Data Sources" to add a new connection for this project.

BuildItem.s	BuildItem.sqlherogen* Start Page									
Dreview	💫 Add Template	e Manage (Data Sources	U History						
Project name:	RD Unit Testing				SQL-Hero application serve	localhost				
Description:	Database tests									
	Log generated	content		Append to log	Halt on error)nlv Log Errors / Warnings				

We've added a single connection which will show up as "Development" for this project:

SQL-Hero Whitepaper Series Codex Enterprises LLC Copyright © 2010

Data Sources usually represent your application as a whole. The active configuration is used when a build is invoked. This allows you to have different connections when building for Debug versus Release, for example. If you are not concerned with the active configuration name can be left blank. Connections are derived from the global connections exposed from SQL-Hero. Having both source and target connections may not always be necessary, depending on the templates being used. Machine name is optional and allows one to have different connections used depending on where the build is being run from. Drag a column header here to group by that column. DataSourceName ConfigName MachineName SourceConnection TargetConnection							
DataSourceName	ConfigName	MachineName	SourceConnection	TargetConnection			
► Development			RD-Development	RD-Development 👻			
*							
			0	K Cancel			

The SourceConnection and TargetConnection must be global connections published in SQL-Hero. You can optionally fine-tune connections based on the current Visual Studio configuration and machine on which the project is being run.

Next, we're going to add a new template to this project:

1	BuildItem.sqlherogen* Start Page									
	Preview	🚯 Add Template	🔮 Manage Data Sources 🔟 History							
	Project name:	RD Unit Testing								
	Description:	Database tests								

We can configure this template as seen here:

Item Detail			×					
🔄 🛃 Save 🛛 🗐 Cancel	E	dit Object List						
Enabled		Sequence: 1	🗖 Run parallel					
Item description:	Unit te	sting template						
SQL-Hero template:	SQL-Hero template: Performance and Unit Test SQL (P, V, U)							
	□ St	ore absolute path	Template Parameters					
Data source:	Devel	opment 💌						
Input man	lager:	By Object Type (Standard)	_					
Source control man	ager:	None	▼					
Output man	ager:	Execute SQL (with result messages)	▼					
		Input Object(s)						
Exclude Pattern =	^\[?(?:(codexspecific history sqlhero)\]?\.						

You must also click on the Template Parameters button since we need to be sure the testing script in this case returns a result set:

Template Paramete	ers (RD-Development)	
Table Name		
Test Types		Minimize
Select Results		
🔲 In Outer Transactio	n	
<u>0</u> K		Cancel

If you click on Save too soon, you'll see something like this:

Input manager:	By Object Type (Standard)	▼
Source control manager:	None	•
Output manager:	Execute SQL (with result messages)	▾ 🔾
		Set control parameters.
	Input Object(s)	bee control parameter of

This is telling us we need to set additional output manager parameters as well, by clicking on the "..." button:

Control Class Parame	ters X
IsLastTable	
ObjectNameColumn	ObjectName
ErrorColumn	ErrMessage
WarningColumn	
MachineName	
4	1
	<u>O</u> K <u>C</u> ancel

Toggling the IsLastTable checkbox to "checked" also sets the ObjectNameColumn and ErrorColumn to defaults which are applicable here. Continue by clicking OK on this screen.

Now we're done with setting up this template, so click Save:

Item Detail	
Save 🧐 Cancel	🛃 Edit Object List
Enabled	Sequence: 1
Item description:	I Init testing template

We're now ready to try running the test project, by building it:



Notice that in this case, we ended with a build failure and an error is showing, right within Visual Studio:



The error in this case is from a failed unit test. Here it makes mention of an invalid object "Customer" and as you can see, there is no Customer table listed in the development database yet. The test has successfully alerted us to a problem. Double-clicking on the error row brings up the detailed error log from the unit test:

```
      RD.DBTests.log
      SQL Hero SQL Editor
      BuildItem.sqlherogen
      Start Page

      -----INFO:
      Starting step 'Unit testing template' at 4/3/2010 3:35:19 PM.
      -----INFO: Object 'PK_Customer_958B614C00551192' of type 'PK' was skipped since template does not append 'UnitPerfTestResults'

      -----INFO:
      Dropped 'UnitPerfTestResults'
      -----INFO: Dropped 'UnitPerfTestResults'

      -----ERR:
      OBJECT:
      [DBO].[UP_SOMECHANGEPROC];
      MESSAGE:

      -----INFO:
      Ending step 'Unit testing template' at 4/3/2010 3:35:20 PM.
```

Yet another place that unit testing can be applied is during the scripting of schema changes from one database to another, using the Schema Compare tool:

🕐 SQL-Hero	
Editor Data Compare Schema Compare Differences History Notifications Crea	ate Data Testing Region Diffs 🛛 🗢 🖡 🕨
🕨 Compare Objects 🔯 Create Script 🙀 Toggle Selection 🧾 Select List For Current Only A	For All Selected 🍸 Filter 🍳 🏢 🌗
Objects	Source: RD-Development
	Target: RD-QA
	Objects
	✓ Tables ✓ Views
	Stored Procedures V Triggers
	User-defined Functions
	Primary Keys V Indexes
	✓ FK Constraints Statistics
	Constraints (other) / defaults
	Extended props, permissions
	Schemas, UDT's, etc.
	Names must match (keys, indexes, etc.)
	Less restrictive matching
	Less restrictive scripting
	Script table diffs as DROP/CREATE
	✓ Ignore case
	Script even if "same"
	Provide a rollback script
	Transactional script
	✓ Include unit tests at end Color legend
	Filter
	C In Both, Same C In Both, Different

Here we produce unit testing script (either in the main script or in a different script if transactional scripting is selected) that could be used to test the scripted objects in the target environment. This assumes the test parameters will be valid there as well, so you do need to consider test results here, case by case. However, it is generally a great way to validate a successful migration of changes.

Test Reports

The Testing tool offers a number of different reports that relate to unit testing.

👯 SQL-Hero	
Editor Data Compare Schema Compa	are Differences History Notifications Create Data Testing Region Diffs
Report: Session History	RD-Development 🔹 🕨 Execute 🍸 Filter 🧰 Schedules 🛍 Apply Tra
Most Recent Durations Recent Errors	/_/ • to _/_/ •
Session History	😰 🗐 🗐 💭 🛃 🖌 📔 100% 🔹 🔹 Find Next
Average Time vs. Targets Averages Over Time (Graph)	
This repor given data Unused Procs	applied traces) recorded for the
Auto Create Proc Tests Object Test History Code Coverage - Chart	<u>otal</u> ≑ <u>Error</u> ≑ <u>Failed</u> ≑ <u>Session Result</u> <u>Dunt Count Target</u> <u>Count</u>
4/2/2010 4:02:02 DM_Multiple	2 4 4 European

Most reports filter to the database listed next to the report type and parameters specific to the report are sometimes available.

The first report we'll examine is "Session History." This report simply lists all occurrences of scheduled testing that have been recorded in the repository. In the example below, we see that the last testing run tested 3 objects, 1 object failed testing, and 1 failed to meet its performance target. (The "session result" is for the session as a whole and is intended to report problems that might have prevented the testing from completing properly.)

Session Hist	tory							For RD-Development
This report lists all tests (null n given database.	un, test run, select fron	n and applied traces)	recorded for the					
Session Date 💲	<u>Type</u>	<u>Total</u> <u>Count</u> €	Error Count	<u>Failed</u> <u>Target</u> <u>Count</u>	\$	Session Result	Only Errors	<u>Username</u>
4/3/2010 4:02:02 PM	Multiple	3	1		1	Success	False	CODEX07\Administrator
4/3/2010 3:59:41 PM	Multiple	2	1		0	Success	False	CODEX07\Administrator
4/3/2010 3:56:21 PM	Multiple	2	1		0	Success	False	CODEX07\Administrator
4/3/2010 8:53:58 PM								1 of 1

4/3/2010 8:53:58 PM

The "Recent Errors" report shows test failures for objects. If "Last Session Only" is checked, it will only consider the very last testing session. If unchecked, an object that failed its test in an older session could end up being shown if it did not later pass its test. In the example below, we see the problem we'd introduced earlier: the Customer table is not present in the development database currently. The object name is hyperlinked such that clicking on it will navigate you to the current version of it in the SQL-Hero editor, using the RD-Development database.

Last Ses	sion Only	_ v to _/_/ v									
i 🖬 M 🛛	🛭 1 of 1 🕨 🕅 🖷 🕄 🛃	3 □ □ □ ↓ 100% Find Next									
Recen	Recent Errors For RD-Developmen										
This report show time-frame. Obj	is only cases where errors were encountered (by objects that no longer exist or are marked as "do not o	ect) during the most recent test run, in a given ver" are not shown on this report.									
<u>Schema</u> 🖨	Object ≑	Error 💠	<u>Ran</u> ≎ <u>Last Mod By</u> <u>On</u>								
dbo	<u>up SomeChangeProc</u>	Invalid object name 'Customer'.; Parameters used: '0HTGE14QWE2', 154119276,NULL,1	4/3/2010 CODEX07\Administrator, 4/3/20 4:02:13 PM 3:35:02 PM, CODEX07	/10							
		Object Count: 1									
4/3/2010 9:05	:01 PM			1 of 1							

The "Most Recent Durations" report shows how long tests took for objects. Note that up SomeSlowProc had a target time of 2,500 milliseconds, so this last execution that took 10 seconds is showing as being 400% of the target. Repeating values are suppressed in the "Test" and "Ran On" columns.

Null Ru	in Call		C Or	ly with Targets			
i 🖬 M 🛛 🖣	1 of 1 🕨 🕅 🗢 🙁 🛃 🗐 🕻	🔲 🔙 🗸 🕴 100	%	•	Find N	ext	
Most R	ecent Durations						For RD-Development
This report show:	the most recent test results for objects in a given database.						
Schema 💲	<u>Object</u> ≑	<u>Time</u>	<u>%</u> Target	<u>Test</u> ≑	<u>Ran</u> ≑ <u>On</u>	Executed SQL	Last Mod By
dbo	up SomeChangeProc	3		Test Run Call	4/3/2010	'0HTGE14QWE2',154119276,NULL,1	CODEX07Administrator, 4/3/2010 3:35:02 PM. CODEX07
	up_SomeSlowProc	10,000	400.00			NULL	CODEX07/Administrator, 4/3/2010 3:58:10 PM. CODEX07
4/3/2010 9:08	IZ PM						1 of 1

4/3/2010 9:08:17 PM

The "Code Coverage" report lists all testable objects in a database and describes what kind of testing is currently available for each object. Object already marked as "do not test" are not shown here. A coverage type of "None" implies the object is not covered by any test type. "Error" indicates the most recent test success/failure status for the object. Clicking on the "Hide" link allows you to mark an object as "do not test" directly from this report.

		// v to _/_/ v									
: 📰 M -	E I < 1 of 1 > I < 3 2 3 I / 100% Tind Next										
Code This report show test run covera	Code Coverage For RD-Development										
<u>Type</u> 🜲	Schema 👙	<u>Name</u> ≑	Coverage Type	≑ <u>Error</u> ≑	Current 💲	Hide					
U	dbo	CustomerType	Select From	False	True	<u>Hide</u>					
P	dbo	up SomeChangeProc	Test Run Call	True	True	<u>Hide</u>					
P	dbo	up SomeSlowProc	Test Run Call	False	True	<u>Hide</u>					
		Covered: 0, Uncovered: 3									
4/3/2010 9:11	1:13 PM					1 of 1					

"Average Durations vs. Targets" shows you an aggregated total of historical test execution times, by object. The report can be filtered to only show objects that have targets (as illustrated below). "Average Over Time (Graph)" is similar in concept, but portrays the average duration on a line graph.

Null Run Call	× /_/	• to _/_/_	• •	Only with Targe	ts						
I II 4 4 1	E M ◀ 1 of 1 ▶ M ← ③ 2 A B I A F 100% ▼ Find Next										
Average D	Average Durations vs. Targets For RD-Development										
This report aggregates d target times, by object.	This report aggregates durations from one or more tests (in a time range) and compares to target times, by object.										
<u>Schema</u> ≑	<u>Name</u> ≑	Avg ‡ Dur.	Max ‡ Dur.	Min Dur. 💲	StdDev \$	<u>Target</u> 韋	<u>%</u> <u>Target</u>	<u>Count</u> 💲	Over Thresho Id		
dbo	up SomeSlowProc	10,000	10,000	10,000		2,500	400.0	1	1		
	Averages / Totals:	10,000.0	10,000	10,000		2,500.0	400.0	1	1		
4/3/2010 9:14:13 PM									1 of 1		

The "Resource Usage Analysis" report attempts to group by a selected facet (in the example below, it's object name) and summarize comparatively within that facet. This could help pinpoint members of a given facet that are consistently out of alignment with the performance of other members. Below, the first procedure is taking up 99.97% of the cumulative duration during testing – if this were part of a more realistic data set, it would be an obvious concern!

	🔶 🔕 🛃	🔲 🛍 🛃 •	100%	•	Fir	nd Next				
Resource Usage For RD-Developm										
This report analyzes duration information by diff	ferent possible groupings.									
¢	<u>Count</u> \$	<u>Total</u> 🖨	Avg \$	<u>% Cnt</u> 💲	<u>% Dur.</u> 💲	<u>DC</u> ≑ Ratio	Max \$	Std. Dev.	Over Dev.	
dbo].[up_SomeSlowProc]	1	10,000	10,000.0	25.00	99.97	3.9988	10,000		0 (+1D), 0 (+2D), 0 (+3D	
dbo].[up_SomeSlowProc]	1 Slowest Call: [di	10,000 bo].[up_SomeSlow	10,000.0 /Proc] NULL	25.00	99.97	3.9988	10,000		0 (+1D), 0 (+2D), 0 (+3D	
dbo].[up_SomeSlowProc] dbo].[up_SomeChangeProc]	1 Slowest Call: [dl 3	10,000 bo].[up_SomeSlow 3	10,000.0 (Proc] NULL 1.0	25.00 75.00	99.97 0.03	3.9988 0.0004	10,000	1.7	0 (+1D), 0 (+2D), 0 (+3D 1 (+1D), 0 (+2D), 0 (+3D	
dbo].[up_SomeSlowProc] dbo].[up_SomeChangeProc]	1 Slowest Call: [dl 3 Slowest Call: [d	10,000 bo].[up_SomeSlow 3 bo].[up_SomeCha	10,000.0 (Proc] NULL 1.0 ngeProc] '0HTGE	25.00 75.00	99.97 0.03 9276,NULL,1	3.9988 0.0004	10,000 3	1.7	0 (+1D), 0 (+2D), 0 (+3D 1 (+1D), 0 (+2D), 0 (+3D	

"Unused Procs" is an interesting report because it can look at test data, trace data, and change tracking data. It can be used to help identify procedures that are not being used. In the example below, we see two objects – these have had no trace data collected and stored in the repository for them. If we checked the "Tests + Traces" checkbox and re-ran the report, no data would come back because we do have tests for them. This report should only be considered an aid in determining what object may no longer be used by an application.

Not changed since:	/_/ Vot traced since: /_/_ V	Tests + Traces									
i 📰 4 🔌 1	of 1 🕨 🕅 🗢 🔕 🛃 🎒 🗐 💭 层 -	100% •		Find Next							
Unused Obj This report lists all stored pro since a date of choice, and he	Inused Objects For RD-Developmen s report lists all stored procedures that have not had a trace (or test) recorded against it ce a date of choice, and has not been modified since a date of choice.										
Schema 🔺	Schema ≜ Name ≜ Create Date ≜ Last Mod Date ≜ Last Mod By ≜										
	<u>Name</u> 🗧				Last mod Date	Last Mod By					
dbo	up SomeChangeProc			4/2/2010 11:44:03 PM	4/3/2010 3:35:02 PM	CODEX07\Administrator					
dbo	vp_SomeChangeProc up_SomeSlowProc			4/2/2010 11:44:03 PM 4/3/2010 3:58:10 PM	4/3/2010 3:35:02 PM 4/3/2010 3:58:10 PM	CODEX07\Administrator					

"Object Test History" reviews the complete testing history for a specific object (or objects matching a portion of a name). One can optionally interleave the change history for the object as stored in the repository. Here we observe the three tests run that included up_SomeChangeProc and the fact that all three tests failed with "Invalid object name 'Customer'".

up_SomeChangeProc	☐ Include chang	e history	🗖 Span all	databases 🔽 I	Exact name match				
i ⊡ 4 4 1	of 1 🕨 🗏 🌲 🕄 😰 🚔 🗐 📜 属	· 100%	, •	Find Next					
Object Test	/ Performance History								
This report lists all past test r	runs for an object, providing the results by date.								
Schema	Name	Туре	Database 💲	Test 💲	Date 💲	Duration	% Target	Message	<u>SQL</u>
dbo	up SomeChangeProc	P F	RD-Development	Test Run Call	4/3/2010 4:02 PM	3		Invalid object name 'Customer'.	'0HTGE14QWE2',154119276,NULL,1
					4/3/2010 3:59 PM	0		Invalid object name 'Customer'.	"0HTGE14QWE2",154119276,NULL,1
					4/3/2010 3:56 PM	0		Invalid object name 'Customer'.	'0HTGE14QWE2', 154119276, NULL, 1

An alternate way to run this report is from the SQL-Hero editor. There is the "Show Testing History" context menu option available on the object tree:

SQL-Hero Whitepaper Series Codex Enterprises LLC Copyright © 2010



The "Code Coverage – Chart" shows a pie chart representation of how many objects have tests, do not have tests yet, or are marked as "do not test." The report can be run for the named database or for *all* global connections available.



Finally, the "Auto Create Proc Tests" report is available to not just provide output, but actually attempt to create unit tests for objects that are missing them. (There is an option to ignore existing tests and try to create all new tests.) In the example below, we've stipulated "5" parameter combinations to try. As seen in the report, we managed to successfully create a new test for up_SomeChangeProc. On the

other hand we could not create a test of up_SomeSlowProc – the error message that prevented creating a test is shown (if there are different messages involved with different test parameters, the first five messages are included). We're also given the option here to "Hide" the object in the future, which means we are marking it as "do not test." Note that while the test creation is running, a button becomes available listing the object being processed – clicking the button effectively terminates the attempt at creating a test. This can be useful if the process becomes "stuck" on a long-running object.

5	✓ Overwrite all existing tests				
i III 4 4 1	of 1 🕨 🕅 🗢 🔕 🛃 🎒 🔲 🔔	- 100% •	Find Next		
Auto Create Tests For RD-Development This report locates procedures that are missing the TestRunParameters UDP and attempts to automatically create test run parameters for these. Objects which failed to have a test created are highlighted in red. This report will not set target times					
Schema 韋	<u>Name</u> ≑		Parameters / Errors	Duration	Hide
dbo	up SomeChangeProc	'YXJF6WETGAEIL6NHAZ3V1',1902322775,1,1		47	<u>Hide</u>
	up_SomeSlowProc	Cannot insert the value NULL into column 'CustomerTypeDesc', table 'RD1.dbo.CustomerType'; column does not allow nulls. INSERT fails.			<u>Hide</u>
4/3/2010 9:38:32 PM					1 of 1