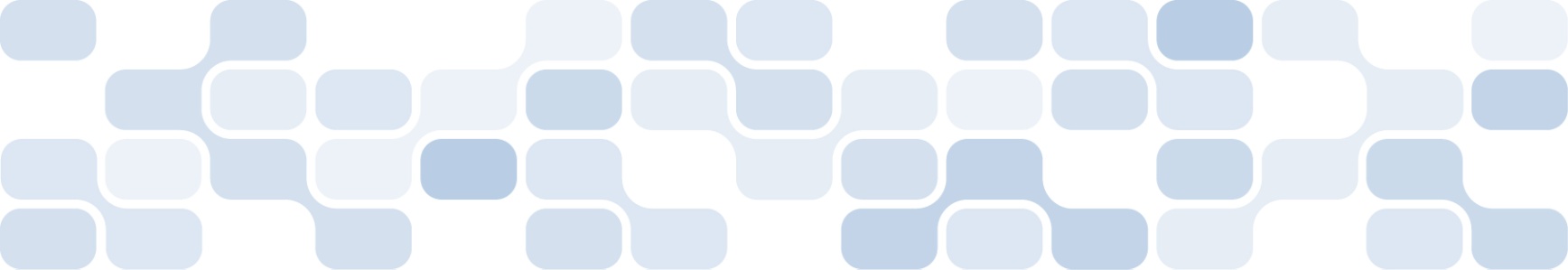


Managing Database Change with Visual Studio® Team System

White Paper

For the latest information, please see [www.microsoft.com/vstudio](http://www.microsoft.com/teamsystem)

****

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, Excel, MSDN, SharePoint, SQL Server, Visual SourceSafe, Visual Studio, and the Visual Studio logo are trademarks of the Microsoft group of companies.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

Introduction 1

Collaborate and Communicate 2

Drive Predictability 3

Manage Team Workflow 4

Use Familiar Tools 5

Ensure Quality, Early and Often 6

Integrate Work Frequently 7

Make Real-time Decisions 8

Summary 9

# Introduction

In the past, many development organizations had people dedicated to handling database issues – writing scripts, creating tables and deploying updates to the database. Today, with fewer developers working on individual projects, that responsibility is, more and more, becoming your responsibility. How do you keep the client-side code in sync with the database code? Even more difficult is remembering all of the scripts you need to deploy to the database for any given release. For many software development projects, you need multiple tools to handle the client code and database code and they don’t work together. Visual Studio® Team System 2008 Database Edition handles all of these issues and more for you and seamlessly integrates into the Visual Studio® Integrated Development Environment.

# The Challenges of Managing Database CHANGE

**The Truth**

Where is your database definition located? In a large majority of organizations, the answer is “in production.” This has some major ramifications. The first is that the only way to get a copy of the current database schema is to request it from the DBA. This takes time and you don’t always get everything you need. Second, if your database is replicated across several servers there is a chance that they are different in subtle ways. And finally, if you ship a shrink wrapped product you may not know what version of a database the customer is using. Each of these issues present problems that are difficult to solve.

The Database Edition of Visual Studio Team System allows you to store the database, at the individual object level, in version control. This helps solve a number of problems. You don’t have to go to the DBA every time you want a copy of the production database. You have a reference copy of the production database available to you so that changes made to individual copies of the database can be easily determined and scripted. And you can go back in time and re-create the database for any particular release by simply retrieving a prior version and performing a build.

**The Impact of Change**

Many of the problems dealing with databases stem from the fact that the database hasn’t been placed under a coherent version control scheme. That is to say that while database objects may be version controlled, they are not aware—of other objects or their relationship to those objects. This presents a difficult situation for making changes. Imagine that you and another developer are making changes, each for your own features, and that those changes in some way relate to each other. How do you know that one developer wrote database code that caused a problem for another developer or vice versa? The short answer is that in today’s environment you can’t really tell. You have to deploy the entire project and run tests which do not happen until many days after the change has been made.

The Database Edition of Visual Studio Team System provides for granular and coherent version control of your database objects. When one developer makes a change to the database, they will immediately see all of the objects which are affected by that change. Likewise, when you check in your changes, you will receive conflicts on just those objects which you are trying to update that have been previously updated by another developer. And afterwards, you will be notified of invalid object relationships. This integrated functionality is unavailable with any other tool on the market.

# Support for Refactoring and Code Analysis

Remember those big projects you worked on where you couldn’t write a line of code before the database design was completely finished? Did you ever wonder why it had to be this way? Of course you did—and you knew the answer: database changes are expensive because they cascade down through the entire application. A database change itself could invalidate numerous other pieces of the database depending on what was changing. Then, the change had to be cascaded to the data access, business logic and the user interface.

The Database Edition of Visual Studio Team System allows you to refactor your database code quickly and easily and helps you validate the effects of database changes. Out of the box, the refactoring tools allow you to preview changes to your database so you know the scope of the change. Then, when your database is refactored, the Database Edition also refactors the related database unit tests and any data access code generated through using datasets. Another benefit of having the database as a project means that you can run tests against changes as quickly as you make the changes. This allows you to pinpoint breaks quickly and fix them with a minimum of effort.

Code analysis of database scripts (stored procedures, functions, triggers, etc.) allows you to enforce code standards or let you know when there may be potential issues with code that’s in the database. This functionality combined with refactoring allows you to easily find and update potentially problematic code.

# Automate Deployment

When building database scripts for release, many times the developers have to send the individual scripts to a DBA so the DBA can construct the final build script. This is a manual process, and with any manual process, mistakes are more likely to occur. Have you ever deployed an update only to find that the application stops working because someone missed a stored procedure?

Because objects in a database solution are broken down to the most granular level, every change to an object is tracked independently. This feature, combined with Team Foundation Server work items, means that you can know a lot more about changes made to the database code. During a build, not only can you generate a SQL script that includes only the changes that were made to the database, but you can *know* what functionality each of the changes applied to! And it isn’t a manual process; it’s automated and because of that it is repeatable.

***General Distribution Release (GDR)***

***Database Edition 2008 – New Features***

*No need for a design-time database*

*Provider (plug-in) model for databases*

*Database / Server project separation*

*Partial and Composite projects*

*Support for SQL Server 2008*

*Separation of build and deployment processes*

*Refactoring, Preservation of Intent*

*Refactoring extensibility*

*Code Analysis extensibility*

*Numerous data generation updates*

But sometimes even this isn’t enough. Take a situation where you have multiple versions of a database deployed in different environments and you need to upgrade all of them to the same version? In the past this required an immense amount of work to customize the release script for each different version of the database. And in today’s security conscious environment, you generally don’t have access to the production database to compare schemas in order to generate the release script. The Database Edition handles this situation as well.

# Summary

With the Database Edition, you can create a reference schema to give to the DBA. Using an included command line tool, the DBA can either apply the schema update to a given database (the tool will determine the differences and generate a script on the fly and apply it to the database) or output a script that the DBA can apply manually. This solves several sticky issues. The DBA has some control over how updates are applied to the database without having to spend a large period of time generating updates scripts, which saves everyone time. Also, the update can be applied against each copy of the database to bring each version up to date with the latest version.

For more information, visit the Web site at <http://www.microsoft.com/defyallchallenges/teamsystem>.