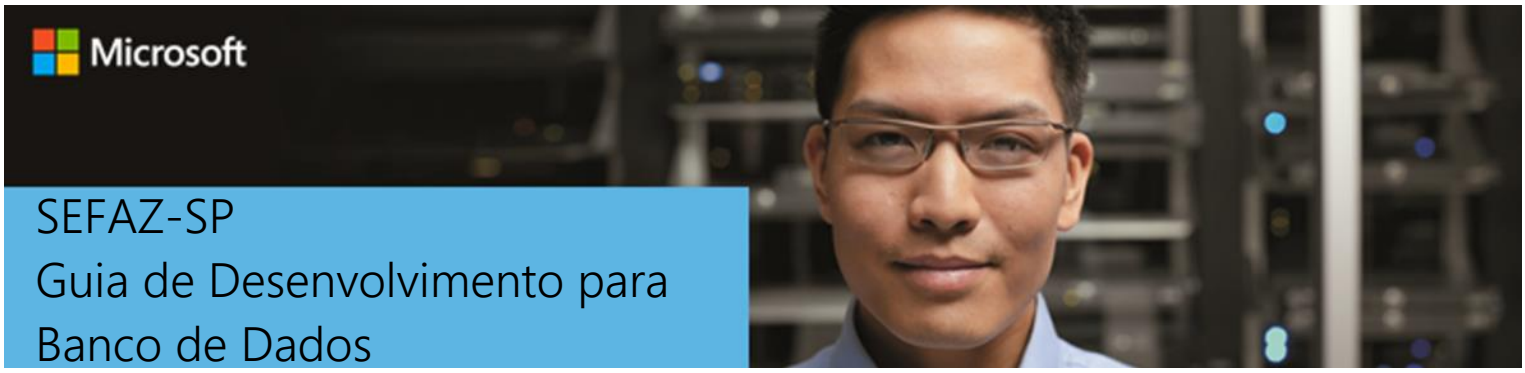

SEFAZ-SP
Guia de Desenvolvimento para Banco de Dados

Preparado para

Secretaria da Fazenda do Estado de São Paulo
segunda-feira, 9 dezembro 2013
Versão 1.0 Final

Preparado por

Regis Gimenis
regis.gimenis@microsoft.com

Revisão

Histórico de mudanças

Data	Autor	Versão	Descrição
15/01/2013	Regis Gimenis	0.1	Versão inicial
23/01/2013	Regis Gimenis	0.2	Recomendações
24/01/2013	Regis Gimenis	0.3	Recomendações
29/01/2013	Regis Gimenis	0.4	Recomendações
05/02/2013	Regis Gimenis	0.5	Revisão
06/02/2013	Gleisson Bezerra	0.6	Revisão final
19/03/2013	Gleisson Bezerra	1.0	Revisão e entrega final do projeto

Índice

Índice de Figuras

Índice de Tabelas

Objetivo

Este documento é parte integrante do projeto de **Elaboração de Arquiteturas de Software – FASE 1**, desenvolvido segundo o Escopo de Trabalho e Formato de Entregas especificados no documento de Visão Escopo (SEFAZ - ITAP Arquiteturas de Software - Visão e Escopo v1.1).

O projeto visa estabelecer uma referência para o desenvolvimento padronizado de sistemas que permita o planejamento da adoção de práticas para reduzir riscos de segurança e aumentar a disponibilidade de serviços da instituição.

Dentro desta proposta foram priorizados os aspectos referentes as seguintes frentes de trabalho:

- **PLATAFORMA .NET**: Diretrizes para organização e construção de componentes que devem servir todas as arquiteturas específicas;
- **SERVIÇOS WEB**: Arquitetura específica, hospedagem e monitoramento de operações;
- **PROCESSAMENTO BATCH**: Arquitetura para aplicações responsáveis por processamentos em lote de forma assíncrona;
- **BANCO DE DADOS**: Revisão de práticas de desenho físico e ferramentas de apoio.

Não foram abordados, nesta fase, a camada de apresentação de **Aplicações, *Providers* de Acesso a Dados e Mapeamento Objeto-Relacional**. Assuntos reservados para as fases seguintes.

Este guia diz respeito à frente de trabalho de Serviços WEB e tem como objetivo é estabelecer práticas relacionadas a modelagem e desenvolvimento de banco de dados na plataforma Microsoft SQL Server, a fim de estabelecer um padrão dentro da Secretaria da Fazenda do Estado de São Paulo (SEFAZ-SP).

O guia irá apresentar um conjunto de práticas recomendadas sobre os seguintes assuntos:

- Ferramenta de desenvolvimento
- Modelagem física de dados
- Codificação T-SQL

Não é objetivo do documento definir o processo de desenvolvimento considerando revisão de papéis e atividades dos envolvidos no desenvolvimento de banco de dados.

Público Alvo

O público alvo deste guia inclui as equipes de desenvolvimento:

- Equipe de desenvolvimento de banco de dados com conhecimento de T-SQL e uso do Visual Studio.
- Desenvolvedores da Coordenadoria de Desenvolvimento de Sistemas e Fábricas de Software com conhecimento de T-SQL e uso do Visual Studio.

Requisitos de Ambiente

Este capítulo apresenta os requisitos de hardware e software.

Requisitos de Software

A ferramenta SSDT recomendada neste guia de desenvolvimento de banco de dados é compatível com os seguintes servidores de banco de dados:

- Microsoft SQL Server 2005 SP4
- Microsoft SQL Server 2008 SP1
- Microsoft SQL Server 2008R2
- Microsoft SQL Server 2012
- Windows Azure SQL Databases

A ferramenta SSDT deve ser instalada nas estações de desenvolvimento com os seguintes requisitos de software:

Estações com Microsoft Visual Studio 2010**:**

- Visual Studio 2010 Service Pack 1
- Windows Vista SP2 ou superior
- Windows 7 SP1 ou superior
- Windows Server 2008 SP2 ou superior
- Windows Server 2008 R2 SP1 ou superior
- SQL Server Data Tools (SSDT) (última atualização)

Estações com Microsoft Visual Studio 2012:

- Windows 7 SP1 (x86 e x64)
- Windows 8 (x86 e x64)
- Windows Server 2008 R2 SP1 (x64)
- Windows Server 2012 (x64)
- SQL Server Data Tools (SSDT) (última atualização)

Observação: A SEFAZ-SP possui algumas estações de desenvolvimento com Windows Server 2003, o SQL Server Data Tools não é suportado nesta versão do sistema operacional.

Segue descrição dos componentes que dão suporte ao desenvolvimento de banco de dados:

Componente	Descrição
SQL Server Data Tools (SSDT)	O SQL Server Data Tools (SSDT) é um conjunto de ferramentas que habilita o profissional de banco de dados bem como o desenvolvedor a realizar o trabalho de modelagem de banco de dados para SQL Server e SQL Azure dentro do Visual Studio. Ele fornece uma interface rica de para desenvolvimento SQL Server com integração completa com Visual Studio, como também ferramenta baseada em modelo que pode ser utilizada para desenvolvimento online e off-line.
SQL Server Data-Tier Application Framework (DACFx)	O Microsoft® SQL Server Data-Tier Application Framework (DACFx v.3.0) é um componente que fornece serviços de ciclo de vida de aplicativos para desenvolvimento de banco de dados e gerenciamento para SQL Server e Windows Azure SQL Databases.

	A Microsoft recomenda usar a versão mais recente disponível da Estrutura de Aplicativo da Camada de Dados (DACfx versão 3.0 ou superior). A Microsoft recomenda usar a versão mais recente disponível dos projetos de banco de dados do SQL Server Data Tools.

Observação: Os projetos de bancos de dados do Visual Studio 2010 não dão suporte a pacotes DACPAC do DAC 3.0 ou pacotes de exportação DAC (BACPAC) gerados com o DACfx versão 3.0 ou posterior.

Requisitos de Hardware

Seguem os requisitos mínimos de hardware para o Visual Studio 2010:

- Processador 1.6 GHz ou mais rápido
- 1 GB RAM
- 3 GB de disco rígido 5400 RPM
- Vídeo: 1024 x 768

Seguem os requisitos mínimos de hardware para o Visual Studio 2012:

- Processador 1.6 GHz ou mais rápido
- 1 GB RAM
- 10 GB de disco rígido 5400 RPM
- Vídeo: 1024 x 768

Siglas

Segue abaixo a lista de abreviaturas utilizadas neste documento:

Sigla	Descrição
SQL	Structured Query Language
T-SQL	Transact-SQL (Extensão SQL da Microsoft)
SSDT	SQL Server Data Tools
SSMS	SQL Server Management Studio
TFS	Team Foundation Server
DACFx	SQL Server Data-Tier Application Framework
SSDE	SQL Server Database Engine
DAC	Data Tier Application
DACPAC	DAC Package
BACPAC	Backup Package (encapsula o esquema de banco de dados e também os dados)

Tabela 1 Siglas

Visão Geral de Arquitetura

Atualmente o desenvolvimento de banco de dados na SEFAZ-SP é realizado diretamente em um servidor Microsoft SQL Server, no qual o desenvolvedor trabalha conectado. Pela ferramenta SQL Server Management Studio o desenvolvedor faz a criação dos objetos do banco de dados, como tabelas, *views, stored procedures* e outros. Este modelo de desenvolvimento não permite um desenvolvimento integrado da aplicação e de banco de dados, ficando por conta do desenvolvedor o gerenciamento de versões e geração dos scripts para a implantação do modelo e código de banco de dados nos ambientes.

Com a utilização do SQL Server Data Tools o objetivo é integrar o desenvolvimento de banco de dados na ferramenta Microsoft Visual Studio, com uso de projeto do tipo banco de dados. Dessa forma será possível ter a integração do desenvolvimento de banco de dados com o ciclo de desenvolvimento de software da SEFAZ-SP, gerenciamento de versões no Team Foundation Server, possibilidade de criação de testes unitários, realização de análise de código estático e geração do pacote durante o processo de build manual ou automático.

No diagrama abaixo são apresentados os componentes de software envolvidos neste guia de desenvolvimento.

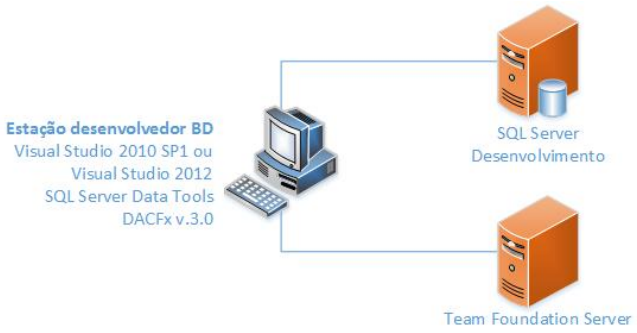

Figura 1 Ambiente de desenvolvimento

A tabela abaixo apresenta os papéis e responsabilidades do fluxo básico de trabalho considerando o uso da ferramenta SQL Server Data Tools e os papéis existentes atualmente na SEFAZ:

Atividade	Responsável	Ferramenta
Criar o projeto de banco de dados	Desenvolvedor	Visual Studio / SSDT
Gerenciar versão do projeto	Desenvolvedor	Visual Studio / TFS
Realizar análise de código estático	Desenvolvedor	Visual Studio / SSDT
Gerar o pacote DAC para implantação no servidor SQL Server de desenvolvimento	Desenvolvedor	Visual Studio / SSDT
Implantar o pacote DAC no servidor SQL Server de desenvolvimento	DBA Desenvolvimento	SSMS

Tabela 2 Papéis e responsabilidades

Observação: A SEFAZ-SP deverá avaliar seu fluxo completo de desenvolvimento de banco de dados, pois durante a análise foi identificado que não existe alguns papéis e atividades que normalmente fazem parte de desenvolvimento de banco de dados, como o papel do arquiteto de dados e gerenciamento de versões de esquema da base de dados.

Recomendações

Neste capítulo são apresentadas as recomendações sobre o desenvolvimento de banco de dados.

Escolha de tipos de dados

O desenvolvedor deverá sempre escolher o tipo de dados apropriado para as colunas da tabela, seguem alguns guias para a definição de tipo de dados:

- Utilizar o tipo de dados de menor tamanho para cada coluna de acordo com a necessidade de negócio.
- Escolher tipos de dados que evitem a conversão de tipos (implícita ou explícita), isto pode ter um custo no desempenho das consultas.
- Evitar a permissão de valores nulos em campos de chaves estrangeiras, isto irá limitar o número de outer joins que precisarão ser escritos.
- Preferir o uso do tipo VARCHAR ao invés do tipo TEXT, o tipo TEXT tem custo adicional porque são armazenados separadamente em páginas de texto/imagem ao invés de páginas de dados.
- Evitar o uso do tipo SQL_VARIANT. Apesar deste tipo permitir flexibilidade pois o campo poderá armazenar valores de diferentes tipos de dados, implica em custo adicional devido a necessidade de armazenamento de metadados e pode ter impacto em desempenho devido a necessidade de conversão de tipos.
- Considerar que os tipos de dados Unicode (NCHAR ou NVARCHAR) ocupam o dobro de espaço comparado ao tipos de dados ASCII (CHAR ou VARCHAR).
- Não utilizar os tipos de dados VARCHAR e NVARCHAR para colunas com tamanho menor que 4 caracteres.
- Utilizar o tipo de dados XML somente nos seguintes casos:
    - Quando dados semiestruturados são necessários.
    - Quando parte de uma entidade não pode ser estruturada com técnicas padrão de modelagem de dados ou
    - Quando parte de uma entidade já existente é um XML e precisa ser armazenado nativamente.
- Considerar a duplicação de campos do XML em colunas da tabela quando parte do conteúdo do XML precisa ser pesquisado, dessa forma é possível criar um índice sobre a coluna.

Campos Identificadores

O desenvolvedor deverá gerar o GUID na aplicação (na camada cliente ou na camada de negócios) quando existir a necessidade de inserção de registro numa tabela que tenha um campo UNIQUEIDENTIFIER. Evitar a geração do GUID pelo SQL Server como propriedade default do campo.

No caso de necessidade de geração de GUID em *stored procedures*, deve ser utilizada a função *newid*() para a criação do valor.

Campos nulos

O desenvolvedor deverá evitar a possibilidade de nulo nas colunas, nulos devem ser aceitos somente quando há um requisito de negócio.

Como prática recomendada, colunas que serão utilizadas como critério de busca não devem aceitar nulo. Se uma coluna de entrada de pesquisa permite nulo, a interface de usuário deve fornecer um meio para indicar isto.

Índices

O desenvolvedor de banco de dados deve analisar e criar os índices das tabelas conforme a previsão de uso pela aplicação.

Apesar do índice ser um mecanismo fundamental para melhorar os tempos de resposta das consultas, deve-se considerar existe um custo associado, tanto de espaço e quanto de desempenho durante a execução de comandos de *insert, update* e *delete.*

Seguem alguns guias para a criação de índices:

- Criar índices baseado no uso.
- Manter índices clustered tão pequenos quanto possível.
- Avaliar a criação de índices nas chaves estrangeiras.
- Criar índice com alto nível de seletividade.
- Criar índices compostos com a coluna mais restritiva em primeiro lugar.
- Considerar a criação de índices sobre colunas utilizadas em clausulas WHERE, ORDER BY, GROUP BY e DISTINCT.

O desenvolvedor de banco de dados deverá criar os índices no filegroup apropriado, ao menos que se trate de uma tabela particionada, neste caso, deve-se criar o índice utilizando o mesmo esquema de particionamento da tabela.

Para campos XML, o desenvolvedor de banco de dados deve considerar a criação de índices XML nos casos abaixo. Deve ser considerado também o alto consumo de espaço de um índice XML (até quatro vezes o tamanho da coluna) e portanto, utilizar somente quando for realmente necessário.

- Quando muitos campos do XML precisam ser pesquisados.
- Existem múltiplas ocorrências num único XML.
- Existe a necessidade de busca pela existência de uma entidade.

Chaves Estrangeiras

O desenvolvedor de banco de dados deve modelar as chaves estrangeiras de forma a não aceitar valores nulos e também deve sempre utilizar a cláusula WITH CHECK. Além de garantir a integridade referencial, isto irá auxiliar o otimizador de consulta para a criação do melhor plano de execução.

UDF (user defined function)

O desenvolvedor de banco de dados deve criar uma User Defined Function somente quando não existir uma função SQL Server equivalente.

UDT (user defined types)

O desenvolvedor de banco de dados deve evitar o uso de User Defined Types (UDT).

Esquemas de Banco de Dados

O desenvolvedor de banco de dados deve criar todos os objetos de banco de dados em um esquema apropriado. O esquema irá auxiliar na divisão dos objetos em agrupamentos lógicos.

Utilizar os seguintes guias para utilização de esquemas:

- Referir a um objeto utilizando seu FQN (*full qualified name*), ao menos utilize o nome do esquema seguido do nome do objeto, separado por um ponto (.).
- Simplificar a implementação de esquemas através do uso de sinônimos para abstrair o proprietário do esquema dos objetos.
- Gerenciar permissões de usuários no nível de esquema. Isto ajuda a proteger os objetos de serem alterados por usuários.
- Utilizar esquemas para combinar entidades relacionadas num único banco de dados físico.

Views

O desenvolvedor de banco de dados deve utilizar *views* para auxiliar no desenvolvimento de banco de dados, principalmente para construir consultas que possuam múltiplos junções entre diferentes tabelas.

Utilizar os seguintes guias para a criação de *views*:

- Evitar o uso de Query Hint numa *View*.
- Sempre analisar o plano de execução de uma *view*.

Views Indexadas

O desenvolvedor de banco de dados deve utilizar o recuso de *view* indexada para aumentar o desempenho das consultas, através da criação de índices para as consultas da *view*.

Utilizar os seguintes guias para a criação de *views indexadas*:

- ANSI_NULLS e QUOTED_IDENTIFIER devem estar configurados como ON no momento de criarmos a indexed *view*.
- A *view* indexada não deve fazer nenhuma referência à outra *view*.
- Todas as tabelas referenciadas nesta *view* devem estar no mesmo banco de dados e devem pertencer ao mesmo schema da *view*.
- A *view* indexada deve ser criada com a opção SCHEMABINDING.
- Tabelas e funções referenciadas na *view* devem especificar o FQN.

Views Particionadas

O desenvolvedor de banco de dados deve evitar o uso de *view*s particionadas.

Stored Procedures

O desenvolvedor de banco de dados deve utilizar stored procedures para o acesso aos dados. A stored procedure deverá conter somente o acesso a dados e não deverá conter regras de negócio, o que deve estar na camada de regra de negócio da aplicação.

Utilizar os seguintes guias para a criação de stored procedures:

- Utilizar parâmetro de saída ao invés de retornar um conjunto de registros quando a stored procedure retornar somente um registro.
- Sempre utilizar a cláusula SET NOCOUNT ON no início da stored procedure;
- Não utilizar comandos PRINT em produção;

- Retornar somente os registros que serão utilizados pela aplicação;
- Retornar somente as colunas que serão utilizadas pela aplicação;
- Evitar o uso do prefixo sp_ no nome da stored procedure.

SQL Dinâmico

O desenvolvedor de banco de dados deve evitar o uso de SQL dinâmico. No caso de não haver alternativa, é preferível utilizar o comando sp_executesql ao invés do comando EXEC ('string'). Isto para evitar risco de SQL injection e permitir o reuso de plano de execução.

Cursores

O desenvolvedor de banco de dados deve evitar o uso de cursores.

Cursores são utilizados para processamento registro a registro de uma tabela. Na maioria dos casos, o mesmo resultado pode ser obtido através de comandos SQL que façam a operação baseado em conjunto de registros.

Caso o uso de cursor seja imprescindível, é necessário garantir o fechamento e desalocação do cursor.

Transações

O desenvolvedor de banco de dados deve manter as transações mais curtas quanto possível.

Isto irá reduzir o número de *locks* sobre os objetos envolvidos na transação e reduzir o risco de *deadlock*.

Transaction Isolation Level

O desenvolvedor de banco de dados deve evitar o uso do nível de isolamento READ UNCOMMITED, seja através do comando SET TRANSACTION ISOLATION LEVEL ou seja através do uso do NOLOCK.

Apesar do READ UNCOMMITTED permitir evitar problemas de bloqueios de registros, ele não é recomendável pois a transação poderá trazer dados inconsistentes.

Preferir o nível de isolamento padrão do SQL Server, READ COMMITTED que proporciona um bom equilíbrio entre consistência e concorrência.

Quando um grande número de leituras precisa ser realizar em tabelas que possam ter alterações, o uso do nível de isolamento SNAPSHOT deve ser considerado. As modificações sobre os dados realizadas por outras transações após o início da transação corrente não são visíveis aos comandos executados durante a transação. As transações SNAPSHOT não requerem *locks* para leitura de dados, não bloqueando outras transações de realizar escritas.

Particionamento de tabelas

O desenvolvedor de banco de dados deve considerar particionar as tabelas nos seguintes casos:

- Para otimizar a performance de consultas em tabelas muito grandes (acima de 10 milhões de registros) ou em tabelas que possuem uma previsão de crescimento grande. A melhoria de desempenho se dá através do acesso a registros somente da partição quando a chave da partição é utilizada na consulta.
- Para facilitar o gerenciamento de tabelas muito grandes, no caso de tabelas precisarem ter a exclusão de um grande conjunto de registros regularmente sem a interrupção de acesso a tabela.

- Para reduzir o tempo de indisponibilidade de um sistema, em situações de desastre.

Considerar as estratégias de particionamento de tabelas e índices do documento "Partitioned Table and Index Strategies Using SQL Server 2008"

Filegroups

O desenvolvedor de banco de dados deve considerar que todo banco de dados deverá ter no mínimo 3 *filegroups*:

- Sistema: contém apenas o esquema do banco de dados e código;
- Dados: contém as tabelas e índices clustered.
- Index: contém todos os índices não clustered.

Quando estiver trabalhando com tabelas particionadas, filegroups extras deverão ser criados, um para cada partição.

Para a uniformização entre os ambientes de Desenvolvimento, Homologação e Produção é necessário seguir o seguinte padrão de criação de FileGroups no ambiente de desenvolvimento:

FileGroups:

FileGroup	Sistema	Primary
FileGroup	Dados	FG_NomeDB_Data_01
FileGroup	Index	FG_NomeDB_Index_01

Padrão de Nomes Lógicos:

FileGroup	Primary	F_NomeDB_Primary_01
FileGroup	Dados	F_NomeDB_Data_01_01
FileGroup	Index	F_NomeDB_Index_01_01
FileGroup	Log	F_NomeDB_Log_01

Padrão de Nomes Físicos:

File Name	Primary	F_NomeDB_Primary_01.mdf
File Name	Dados	F_NomeDB_Data_01_01.ndf
File Name	Index	F_NomeDB_Index_01_01.ndf
File Name	Log	F_NomeDB_Log_01.ldf

Comandos Depreciados do SQL Server 2012

O desenvolvedor de banco de dados deve conhecer e evitar o uso de comando depreciados. Comandos depreciados são comandos do SQL Server que podem não estar disponíveis numa versão futura.

Segue a lista de alguns principais comandos depreciados do SQL Server 2012:

- Column aliases in ORDER BY clause cannot be prefixed by table alias
- COMPUTE clause is not allowed in database compatibility 110
- Deprecated / unsupported system stored procedures
- FASTFIRSTROW table hint usage
- FOR BROWSE is not allowed in *views* in 90 or later compatibility modes
- FOR XML AUTO queries return derived table references in 90 or later compatibility modes
- Non ANSI style left outer or right outer join usage
- Non-integer constants in the ORDER BY clause

- Numbered Procedures are deprecated
- ORDER BY clauses in *view*s
- ORDER BY specifies integer ordinal
- Remove references to undocumented system tables
- SETUSER statement usage
- Subqueries are not supported in GROUP BY clause
- Unqualified Joins
- VARCHAR / NVARCHAR declared without size specification
- WITH CHECK OPTION is not supported in *view*s that contain TOP in 90 or later compatibility modes

Referências

- Partitioned Table and Index Strategies Using SQL Server 2008 (http://download.microsoft.com/download/D/B/D/DBDE7972-1EB9-470A-BA18-58849DB3EB3B/PartTableAndIndexStrat.docx)
- SQL Server Data Tools (SSDT) (http://msdn.microsoft.com/pt-br/library/hh272686(v=VS.103).aspx)
- Get Started with Microsoft SQL Server Data Tools (http://msdn.microsoft.com/en-us/data/hh297027)
- SQL Server Data Tools - December 2012 update (http://msdn.microsoft.com/en-us/jj650015)
- Available Today: SSDT—December 2012 (http://blogs.msdn.com/b/ssdt/archive/2012/12/13/available-today-ssdt-december-2012.aspx)

Apêndice A – SSDT

1 A ferramenta SQL Server Data Tools

O SQL Server Data Tools (SSDT) é um conjunto de ferramentas que habilita o profissional de banco de dados bem como o desenvolvedor a realizar o trabalho de modelagem de banco de dados para SQL Server e SQL Azure dentro do Visual Studio, incluindo criação dos objetos como tabelas, *views* e *stored procedures*.

Ele fornece uma interface rica para desenvolvimento SQL Server, com integração completa com Visual Studio, como também ferramenta baseada em modelo que pode ser utilizada para desenvolvimento online e off-line.

As funcionalidades disponíveis são:

- Criação do esquema da base de dados
- Comparação entre esquemas
- Suporte a desenvolvimento T-SQL (*views*, stored procedures, functions)
- Refatoração
- Testes unitários de banco de dados
- Permite o desenvolvimento de banco de dados conectado ou off-line
- Integração com gerenciador de código-fonte para controle de versões e mudanças.
- Implantação através de scripts que podem ser aplicados imediatamente ou publicados para execução.

O SSDT permite dois modos de trabalho:

- Desenvolvimento de banco de dados conectado, dessa forma o desenvolvedor pode fazer a criação de objetos locais e também diretamente numa instância SQL Server; ou
- Desenvolvimento de banco de dados off-line, para os casos em que o desenvolvedor não tem acesso a nenhuma instância SQL Server. Assim ele cria e edita todos os objetos localmente no Visual Studio e o administrador de banco de dados fica responsável pela publicação dos objetos na instância SQL Server.

Referências:

- http://msdn.microsoft.com/pt-br/library/hh272686(v=VS.103).aspx
- http://msdn.microsoft.com/en-us/data/hh297027

2 Integração do SSDT com o Team Foundation Server

O desenvolvedor de banco de dados deverá gerenciar as mudanças de banco de dados integrado ao Microsoft Team Foundation Server 2010 (TFS 2010).

A integração com o TFS 2010 para projetos de banco de dados é realizada da mesma maneira que outros projetos através de check-in e check-out dos arquivos e através do Team Explorer.

3 Testes unitários de banco de dados

O SSDT permite ao desenvolvedor criar e executar testes unitários de banco de dados interativamente no Visual Studio, ou através de linha de comando ou no processo de build. Testes unitários de banco de dados são recomendáveis para validar o código SQL em ambiente controlado antes da implantação nos ambientes de homologação e produção.

O desenvolvedor de banco de dados deverá criar e executar testes unitários de banco de dados durante o processo de desenvolvimento para garantir a qualidade dos objetos criados ou modificados.

Observação: Testes unitários de banco de dados estão disponíveis somente a partir da atualização do SSDT de Dezembro de 2012.

Referências:

- SQL Server Data Tools - December 2012 update (http://msdn.microsoft.com/en-us/jj650015)
- Available Today: SSDT—December 2012 (http://blogs.msdn.com/b/ssdt/archive/2012/12/13/available-today-ssdt-december-2012.aspx)

4 Realização de análise de código estático

O desenvolvedor deverá realizar a validação de código SQL conforme as regras do Visual Studio. A lista de regras de validação está apresentada abaixo.

Microsoft.Rules.Data.Design

- SR0001: Avoid SELECT * in stored procedures, *view*s, and table-valued functions.
- SR0008: Consider using SCOPE_IDENTITY instead of @@IDENTITY.
- SR0009: Avoid using types of variable length that are size 1 or 2.
- SR0010: Avoid using deprecated syntax when you join tables or *view*s.
- SR0013: Specify values for output parameters in all code paths.
- SR0014: Maintain compatibility between data types.

Microsoft.Rules.Data.Naming

- SR0011: Avoid using special characters in object names.
- SR0012: Avoid using reserved words for type names.
- SR0016: Avoid using sp_as a prefix for stored procedures.

Microsoft.Rules.Data.Performance

- SR0004: Avoid using columns that do not have an index as test expressions in IN predicates.
- SR0005: Avoid using patterns that start with '%" in LIKE predicates.
- SR0006: In the comparison, simplify the expression that includes indexed columns.
- SR0007: Use ISNULL(column, default value) on nullable columns in expressions.
- SR0015: Extract deterministic function calls from WHERE predicates.