SQL Server™ .NET Programmability

José A. Blakeley
Architect
SQL Server

Background

- Transact SQL (T-SQL)
  - SQL Server's database programming language
  - Includes:
    - SQL DDL, SQL DML
    - Variables, assignment, iteration, procedures, functions
- .NET Platform
  - .NET runtime (Common Language Runtime)
    - Verifiable Intermediate Language
    - Managed memory – garbage collection
    - Multiple languages (e.g, C#, C++, Cobol)
  - Frameworks library (e.g., GUI, Files, XML)
  - Assemblies – New Dynamic Link Libraries
    - Contain code + metadata (class definitions, assembly dependencies)
    - Unit of code deployment, security, versioning

Outline

- Motivation
- Basic Infrastructure
  - Hosting overview
  - Data access inside process: ADO.NET
  - SQL types
- SQL features
  - Server assemblies
  - Functions, procedures, triggers
  - Types and aggregates
- Summary

Motivation

- Broaden the set of languages for running data-intensive business logic in server
  - IT Developers and ISVs
  - Multiple languages: Visual Basic®, C#, C++, J#, …
  - Same mid- and server-tier data access model
  - Same tools: IDE, project model, debugging, …
  - Transact-SQL still supported
- Packaged as .NET assemblies
  - Verifiable, secure
- Deployed to SQL Server as
  - Functions, procedures, triggers, types
- Basis for SQL Server Extensibility

Development Steps

Basic Infrastructure

- Hosting common language runtime inside SQL Server
  - 4Ss: Safety, security, scalability, speed
  - Run verified, type-safe code inside process
  - Multiple languages
- Data access in process
  - Based on ADO.NET
  - Same data access programming model as middle-tier
- SQLTypes
  - SQL type semantics in managed code

Hosting .NET Runtime

- Safety
  - Prevent user code from corrupting the server process
    - Verifiable code
  - Use runtime code permissions to control user code when calling
    - Unmanaged APIs, user interface, threads, synchronization
- Security
  - Authorized access to SQL Server data from user code via SQL Server authorization model
  - Authorized access to system resources from user code via runtime code permissions
  - Administrators control permissions given to assemblies

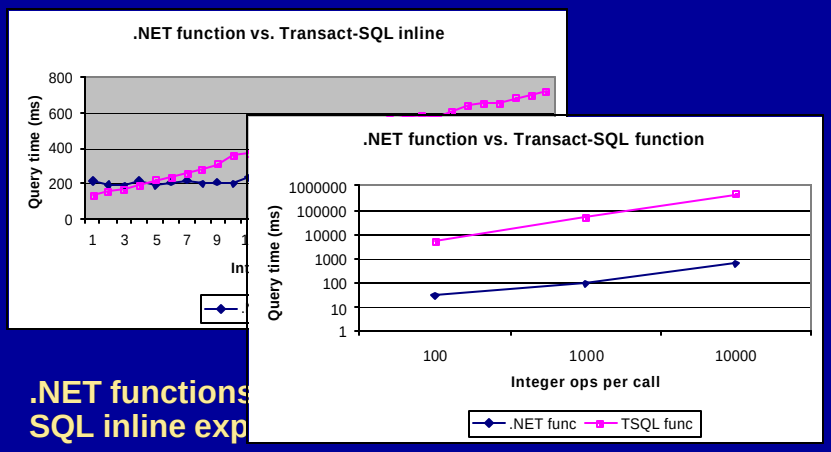
Three Code Permission Sets

- SAFE
  - Execute & data access permission
  - No access to resources outside SQL Server
  - No unmanaged calls
  - Must be verifiable
- EXTERNAL_ACCESS
  - SAFESQL + access to external resources (Net, File permissions)
  - Requires EXTERNAL ACCESS permission to create
  - SQL Server will impersonate the caller
  - Must be verifiable
- UNRESTRICTED
  - No controls: Can call unmanged code, can be un-verifiable
  - Only Administrators can create

Hosting .NET Runtime

- Scalability
  - As many concurrent users as, as fast as Transact-SQL
    - Integrated SQL Server and runtime threads
    - Collaboration between SQL Server and GC
    - SQL Server becomes OS to the .NET runtime
- Speed
  - Efficient data access in process
  - Compiled user code, not interpreted as Transact-SQL
  - Fast transitions in/out of runtime

Speed: Functions

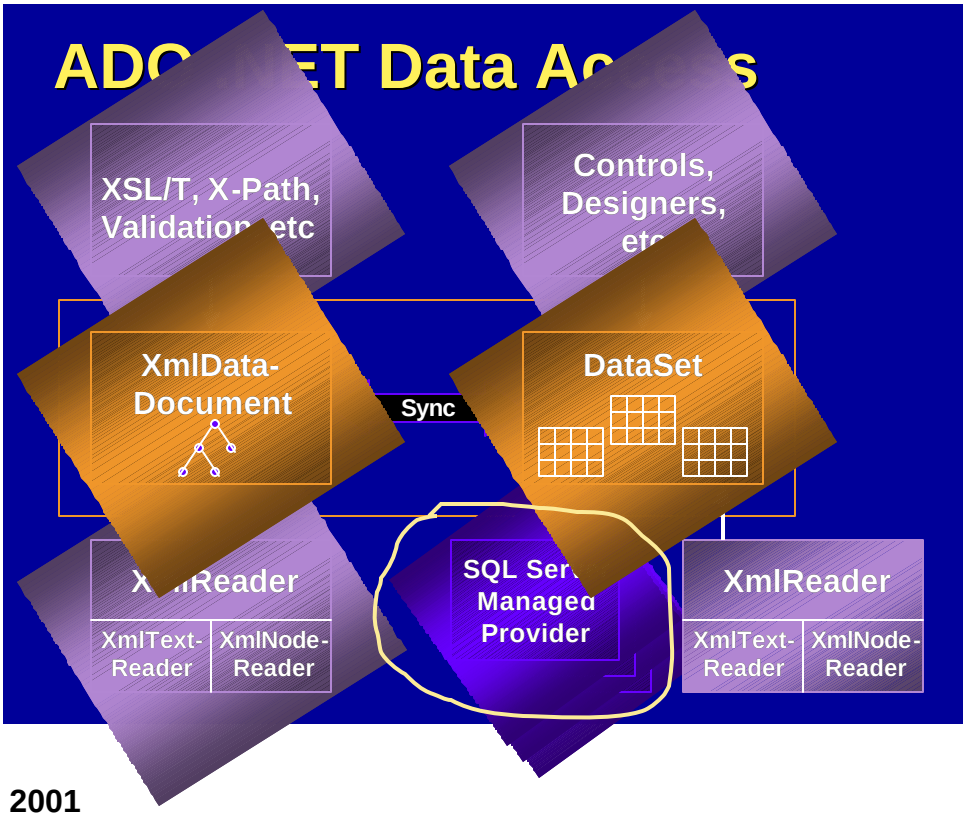


.NET functions
SQL inline exp

.NET framework functions much faster than Transact-SQL functions for complex expressions

Outline

- Motivation
- Basic Infrastructure
  - Hosting overview
  - Data access inside process: ADO.NET
  - SQL types
- SQL features
  - Server assemblies
  - Functions, procedures, triggers
  - Types and aggregates
- Summary



Data Access Inside SQL

```
using System.Data.SqlServer;
using System.Data.SqlTypes;

public class ShippingCosts {
public static SqlMoney FreightByShipper( string ship ) {
    SqlCommand cmd = SqlContext.GetCommand( );
    cmd.CommandText = "select sum(o.freight) as freight " +
    "from orders o join shippers s on o.shipvia = s.shipperid " +
    "where s.companyname = @CompanyName " ;
    SqlParameter param = cmd.Parameters.Add("@CompanyName",
          SqlDbType.NVarChar, COMPANY_NAME_COL_LENGTH);
    param.Value = ship;
    SqlMoney amount = cmd.ExecuteScalar( );
    return amount;
    }
}
```

SQL Types

- Reduce impedance mismatch between programming language and data
- Consistent expression evaluation in mid- and server-tier programming
- SQL Types library
  - Managed classes: System.Data.SqlTypes
  - Provide SQL semantics
    - Nullability, three-valued logic
    - Precision and scale in operations

SQL Types Example

- Tax function using SQL types

```
using System;
using System.Data.SQLTypes;

public class myFinances
{
        public static SQLDouble tax( SQLDouble sal )
        {
        if ( sal < 50000.0 ) return sal * 0.15;
        if ( sal >= 50000.0 && sal <= 90000.0 ) return sal * 0.23
        else return sal * 0.35;
        }
}
```

SQLDouble makes function NULL-aware

Outline

- Motivation
- Basic Infrastructure
  - Hosting overview
  - Data access inside process: ADO.NET
  - SQL types
- SQL features
  - Server assemblies
  - Functions, procedures, triggers
  - Types and aggregates
- Summary

SQL Features

- Assemblies
- Functions
- Procedures
- Triggers
- Types

Creating An Assembly

```
CREATE ASSEMBLY geom FROM '\\m1\types\geometry.dll'
    WITH PERMISSION_SET = SAFE
    WITH AUTOREGISTER
DROP ASSEMBLY lib_geom
```

- Assemblies stored in database
  - Backup, restore with data
- Code permissions assigned per assembly
  - Safe (default), external access, unrestricted
- Autoregister functions
  - Using .NET custom attributes
- Assembly benefits
  - Self-describing metadata: Types, file dependencies
  - Unit of code deployment: Permissions, versioning

Altering An Assembly

- Cannot invalidate persistent data or indexes
- Implies
  - No tables with columns of UDT from this assembly
  - No indexes on functions of this assembly
- Force option allows ALTER even if persistent dependencies exist
- Packaging considerations
  - Place routines and types in different assemblies

Creating A Function

```
CREATE FUNCTION distance (
    @x1 int, @y1 int, @x2 int, @y2 int ) RETURNS float
    EXTERNAL NAME 'geom:CPoint.Distance'
    DETERMINISTIC
    RETURNS NULL ON NULL INPUT
DROP FUNCTION distance
```

- Functions called from queries
  - Can be scalar or table-valued
  - Static class functions
  - Deterministic functions
- Using a function in a query

```
SELECT s.name FROM Supplier s
WHERE dbo.distance( s.x, s.y, @x, @y ) < 3
```

Creating A Procedure

```
CREATE PROCEDURE check_inventory
    EXTERNAL NAME 'events:CInventory.check_level'
DROP PROCEDURE check_inventory
```

- Procedures not called from queries
  - Can contain SQL queries, updates or DDL
  - Can return results directly to client

Creating A Trigger

```
CREATE TRIGGER supplier_event ON supplier
    AFTER INSERT, UPDATE
    EXTERNAL NAME 'events:CNotif.Supp_Event'
DROP TRIGGER supplier_event
```

- Similar to procedures, plus
- Access to inserted, deleted tables

Creating A Type

- Create or drop a type

```
CREATE TYPE Point
    EXTERNAL NAME 'geom:Point'
DROP TYPE Point
```

- Using a type

```
CREATE TABLE Supplier (
    id          INTEGER PRIMARY KEY,
    name        VARCHAR(20),
    location    Point )
```

- Using a type method in a query

```
SELECT s.name FROM Supplier s
WHERE s.location::distance( @point ) < 3
```

Query Optimization

- Automatically gather function statistics
  - Value histograms, execution cost
- Reorder of predicate evaluation
  - Based on execution cost and statistics
- Function indexes
  - Speed up expensive functions
  - Extends computed column indexes and index views
- Implied and residual predicates

Summary

- Richer server programming model
  - Any .NET framework language
  - Same mid- and server-tier data access: ADO. NET
  - Same IDE, project model, debugging, tools
- SQL Server hosting of .NET runtime
  - Safety, security, scalability, speed
- SQL Server .NET features
  - Functions, stored procedures, triggers, types, aggregates

Demos

- Sample code for function, stored procedure, and type
- Assemblies
  - Safe versus external access
- Functions and stored procedures
  - Creation and execution
  - Data access in-process via ADO .NET
- Types
  - Type contract
  - Create type
  - Create table with column of user-defined type
  - Create index on column of user-defined type
  - Insert and query the table