Dynamic Database

Abstract

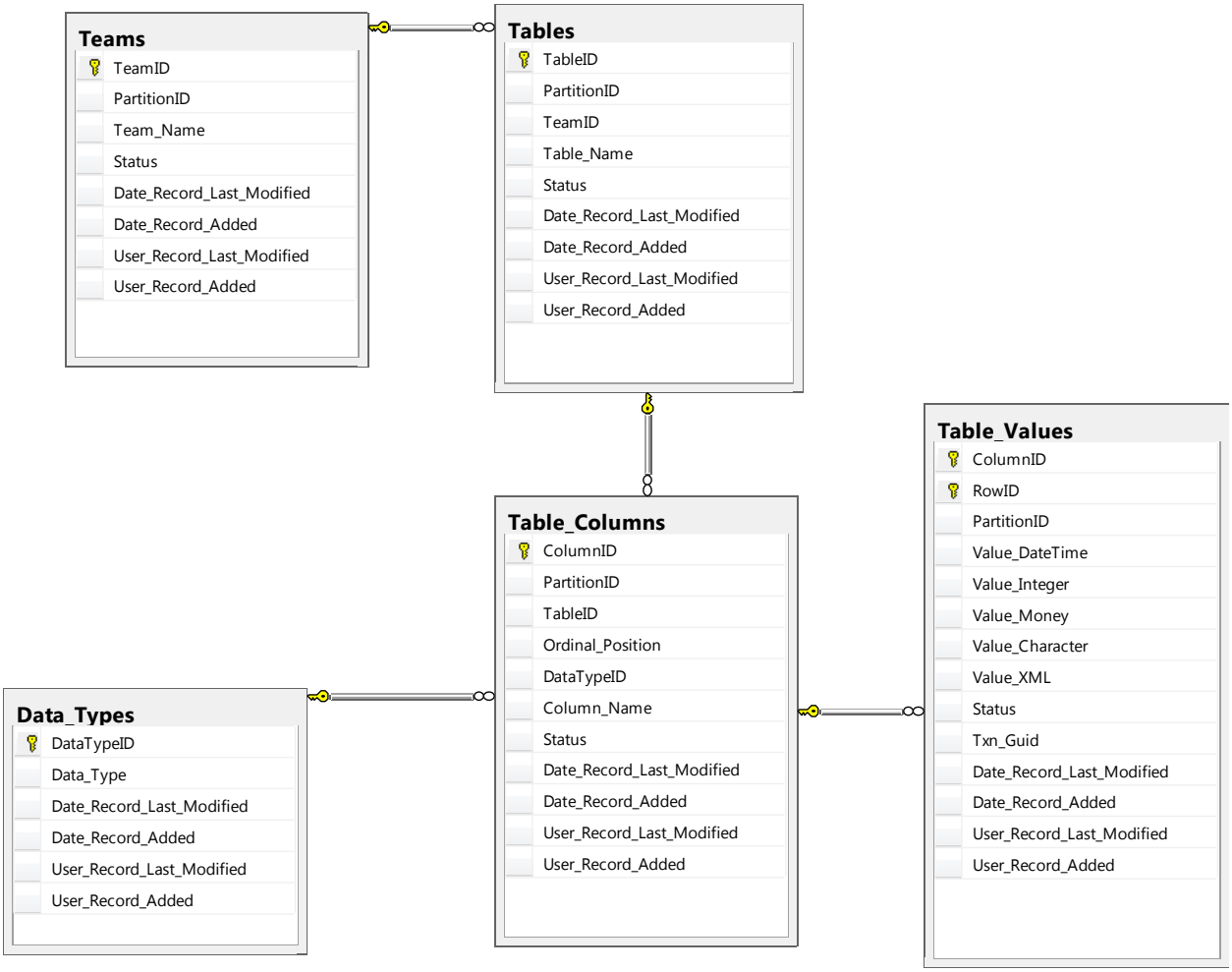
The dynamic database gives a completely data driven schema per team. The goal of this database is to allow small to medium teams to rapidly model and store schemas in an enterprise database, which in turn, requires almost zero additional operation administration as team quantities increase.

Table Design

Design Notes

- Each table has an “Added” and a “Last Modified” column for both user and datetime of change. This is maintained by the database automatically.
- All tables contain a system column called “Status”. This column is used to denote if a row is active or not. It defaults to 1, meaning active, but can be changed to 0 to mean inactive. When the accompanying sprocs are used to maintain data in these tables, deletion of values in these tables is simply a “deactivation” of the row. No data is actually deleted. This is for safety precaution. Should you want to physically remove data there would need to be a maintenance routine developed to delete data “WHERE Status = 0”.
- All “ID” PK columns in each table is an auto incrementing surrogate key, and RFI is enforced by relationships between surrogate keys. Natural keys for each table are defined and enforced using unique indexes.
- Besides unique indexes and PKs, there are no other indexes in this version of the database. Users are encouraged to profile their specific workloads, and add indexes where appropriate for their workload performance.
- All objects in this database are built with page compression. Should you not be able or want to support this, the parameter “DATA_COMPRESSION = PAGE” would need to be removed from the table create scripts.

Logical Model



Schema Definitions

TableName	column_id	ColumnName	Type	Length	PK	Identity	Unique	Description
dbo.Data_Types	1	DataTypeID	int	4	x	x		autoid pk
dbo.Data_Types	2	Data_Type	nvarchar	50			x	text name of data type
dbo.Data_Types	3	Date_Record_Last_Modified	datetime	8				system metadata, date of row modify (getdate()) default
dbo.Data_Types	4	Date_Record_Added	datetime	8				system metadata, date of row add (getdate()) default
dbo.Data_Types	5	User_Record_Last_Modified	nvarchar	255				system metadata, user of row modify (system_user()) default
dbo.Data_Types	6	User_Record_Added	nvarchar	255				system metadata, user of row add (system_user()) default
dbo.Table_Columns	1	ColumnID	bigint	8	x	x		autoid pk
dbo.Table_Columns	2	PartitionID	tinyint	1				system column, may be used as a user defined partitionable value, default 1
dbo.Table_Columns	3	TableID	bigint	8			x	
dbo.Table_Columns	4	Ordinal_Position	int	4				ordinal position of dynamic column
dbo.Table_Columns	5	DataTypeID	int	4				
dbo.Table_Columns	6	Column_Name	nvarchar	100			x	text name of dynamic column
dbo.Table_Columns	7	Status	bit	1				status of record, 1 = active, 0 = inactive, delete functions change status of record from 1 to 0
dbo.Table_Columns	8	Date_Record_Last_Modified	datetime	8				system metadata, date of row modify (getdate()) default
dbo.Table_Columns	9	Date_Record_Added	datetime	8				system metadata, date of row add (getdate()) default
dbo.Table_Columns	10	User_Record_Last_Modified	nvarchar	255				system metadata, user of row modify (system_user()) default
dbo.Table_Columns	11	User_Record_Added	nvarchar	255				system metadata, user of row add (system_user()) default
dbo.Table_Values	1	ColumnID	bigint	8	x			
dbo.Table_Values	2	RowID	bigint	8	x			vertical row number for virtual row
dbo.Table_Values	3	PartitionID	tinyint	1				system column, may be used as a user defined partitionable value, default 1
dbo.Table_Values	4	Value_DateTime	datetime	8				datetime value of virtual row
dbo.Table_Values	5	Value_Integer	int	4				integer value of virtual row
dbo.Table_Values	6	Value_Money	decimal	(19,4)				decimal value of virtual row
dbo.Table_Values	7	Value_Character	nvarchar	max				Unicode character value of virtual row
dbo.Table_Values	8	Value_XML	Xml	max				xml value of virtual row
dbo.Table_Values	9	Status	bit	1				status of record, 1 = active, 0 = inactive, delete functions change status of record from 1 to 0
dbo.Table_Values	10	Txn_Guid	uniqueidentifier	16				system column, transaction identifier for virtual row inserts, used for rollbacks
dbo.Table_Values	11	Date_Record_Last_Modified	datetime	8				system metadata, date of row modify (getdate()) default
dbo.Table_Values	12	Date_Record_Added	datetime	8				system metadata, date of row add (getdate()) default
dbo.Table_Values	13	User_Record_Last_Modified	nvarchar	255				system metadata, user of row modify (system_user()) default
dbo.Table_Values	14	User_Record_Added	nvarchar	255				system metadata, user of row add (system_user()) default
dbo.Tables	1	TableID	bigint	8	x	x		autoid pk

dbo.Tables	2	PartitionID	tinyint	1				system column, may be used as a user defined partitionable value, default 1
dbo.Tables	3	TeamID	int	4			x	
dbo.Tables	4	Table_Name	nvarchar	100			x	text name of dynamic table
dbo.Tables	5	Status	bit	1				status of record, 1 = active, 0 = inactive, delete functions change status of record from 1 to 0
dbo.Tables	6	Date_Record_Last_Modified	datetime	8				system metadata, date of row modify (getdate()) default
dbo.Tables	7	Date_Record_Added	datetime	8				system metadata, date of row add (getdate()) default
dbo.Tables	8	User_Record_Last_Modified	nvarchar	255				system metadata, user of row modify (system_user()) default
dbo.Tables	9	User_Record_Added	nvarchar	255				system metadata, user of row add (system_user()) default
dbo.Teams	1	TeamID	int	4	x	x		autoid pk
dbo.Teams	2	PartitionID	tinyint	1				system column, may be used as a user defined partitionable value, default 1
dbo.Teams	3	Team_Name	nvarchar	100			x	text name of team
dbo.Teams	4	Status	bit	1				status of record, 1 = active, 0 = inactive, delete functions change status of record from 1 to 0
dbo.Teams	5	Date_Record_Last_Modified	datetime	8				system metadata, date of row modify (getdate()) default
dbo.Teams	6	Date_Record_Added	datetime	8				system metadata, date of row add (getdate()) default
dbo.Teams	7	User_Record_Last_Modified	nvarchar	255				system metadata, user of row modify (system_user()) default
dbo.Teams	8	User_Record_Added	nvarchar	255				system metadata, user of row add (system_user()) default

Stored Procedures

Variable Definitions

@Team nvarchar(100): Name of team. Team is analogous to schema, and used in query operations as a security boundary.

@Table_Name nvarchar(100): Name of table.

@Column_Name nvarchar(100): Name of column.

@Data_Type nvarchar(50): Name of data type, and a property of the table value. Currently datetime, integer, character (i.e. nvarchar(MAX)), money (i.e. decimal(19,4)), and xml data types are supported.

@PartitionID tinyint: Partition ID is a means to support table partitioning. This will default to 1, but users may use this column and its value as a means to define table partitioning.

@RowID bigint: The number of the row of a selected result set. Required for all update/delete operations, but also returned as part of the select statements.

@NewVal nvarchar(max): Used in update statements as the new value to populate applicable column. Data type is nvarchar(max), but will be converted to native type of column in flight.

@Top int: Defaults to 0, which means return all rows, but acts as traditional TOP n for select statements. Note that TOP and SQL_Where are not compatible and should not be used together.

@SQL_Where nvarchar(200): Defaults to ‘’, but used as traditional “WHERE” clause for select statements.

@TableSpec AS xml: XML specification for table definition, used in table create statement.

XML Specification

```
<TableSpec>
 <name>table name</name>
  <team>team name</team>
    <Columns>
       <column>
              <name>column name</name>
              <datatype>data type of column</datatype>
       </column>
       <column>
              <name>column name</name>
              <datatype>data type of column</datatype>
       </column>
              </Columns>
</TableSpec>
```

@InsertSpec xml: XML specification for insert statements.

XML Specification

```
<InsertSpec>
<name>table name</name>
  <team>team name</team>
    <Columns>
       <column>
              <name>column name</name>
              <value>value to populate column</value>
       </column>
       <column>
              <name>column name</name>
              <value>value to populate column</value>
       </column>
```

```
    </Columns>
</InsertSpec>
```

Object Definitions

usp_Create_Team

This is for defining a team as a security boundary for access to table data. Before creating tables, an initial team must be created.

Usage

```
DECLARE @Team nvarchar(100), @PartitionID tinyint

SET @Team = 'Rigel';
SET @PartitionID = 1; --Note this is optional, defaults to 1.

EXECUTE [dbo].[usp_Create_Team] @Team, @PartitionID
GO
```

usp_Delete_Team

Deletes a team by setting status bit to 0 in Teams table.

Usage

```
DECLARE @Team nvarchar(100);

SET @Team = 'Rigel';

EXECUTE [dbo].[usp_Delete_Team] @Team
GO
```

usp_Create_Table

Creates a virtual table based on an xml spec. Values for data type are "character" (nvarchar(100), "integer", "money" (decimal(19,4)), "xml" or "datetime".

Usage

```
DECLARE @TableSpec AS xml;
SELECT @TableSpec = '<TableSpec>
 <name>MovieQuotes</name>
 <team>Rigel</team>
    <Columns>
                        <column>
                                <name>QuoteID</name>
                                <datatype>integer</datatype>
                        </column>
                        <column>
                                <name>description</name>
                                <datatype>character</datatype>
                        </column>
                        <column>
                                <name>Published</name>
                                <datatype>datetime</datatype>
                        </column>
                        <column>
```

```
                <name>cost</name>
                <datatype>money</datatype>
            </column>
            <column>
                <name>Entry</name>
                <datatype>xml</datatype>
            </column>
        </Columns>
</TableSpec>';

EXECUTE usp_Create_Table @TableSpec
GO
```

usp_Drop_Table

Drops a virtual table.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100);
SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes';

EXECUTE [dbo].[usp_Drop_Table] @Team, @Table_Name;
GO
```

usp_Add_Column

Adds a column to a virtual table.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100), @Column_Name nvarchar(100), @Data_Type
nvarchar(50);

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes',@Column_Name = 'Lead Actor', @Data_Type = 'character';

EXECUTE [dbo].[usp_Add_Column] @Team, @Table_Name, @Column_Name, @Data_Type;
```

usp_Drop_Column

Drops a column from a virtual table.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100), @Column_Name nvarchar(100);

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes',@Column_Name = 'Lead Actor';

EXECUTE [dbo].[usp_Drop_Column] @Team, @Table_Name, @Column_Name;
```

usp_Show_Schema

Shows definitions of virtual tables.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100);

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes';

EXECUTE [dbo].[usp_Show_Schema] @Team, @Table_Name;

-- For all teams and tables.

EXECUTE [dbo].[usp_Show_Schema]
```

usp_Insert_Data

Inserts a row of data in a virtual table based on an xml spec.

Usage

```
DECLARE @InsertSpec AS xml;
SELECT @InsertSpec = '<InsertSpec>
 <name>MovieQuotes</name>
 <team>Rigel</team>
    <Columns>
        <column>
                <name>QuoteID</name>
                <value>1</value>
        </column>
        <column>
                <name>description</name>
                <value>quote of the day entry</value>
        </column>
        <column>
                <name>Published</name>
                <value>6/4/1982</value>
        </column>
        <column>
                <name>Cost</name>
                <value>99.9999</value>
        </column>
        <column>
                <name>Entry</name>
                <value>   <![CDATA[
                <Best_Movie_Dialog_of_All_Time>
                <title>Star Trek II: The Wrath of Khan</title>
                <Kirk>Spock!</Kirk>
                <Spock>The ship... out of danger?</Spock>
                <Kirk>Yes.</Kirk>
                <Spock>Dont grieve, Admiral. It is logical. The needs of the many outweigh...</Spock>
                <Kirk>...the needs of the few...</Kirk>
                <Spock>...Or the one. </Spock>
                <Spock>I never took the Kobayashi Maru test until now. </Spock>
                <Spock>What do you think of my solution?</Spock>
                <Spock>I have been and always shall be your friend.</Spock>
                <Spock>Live long and prosper. </Spock>
                </Best_Movie_Dialog_of_All_Time>]]>
                </value>
        </column>
```

```
</Columns>
</InsertSpec>'

EXECUTE dbo.usp_Insert_Data @InsertSpec;
```

usp_Select_Data

*Used for "Select *" operations from virtual tables.*

Usage

```
DECLARE @Team nvarchar(100)
DECLARE @Table_Name nvarchar(100)
DECLARE @Top int
DECLARE @SQL_Where nvarchar(200)

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes';
EXECUTE [dbo].[usp_Select_Data] @Team,@Table_Name,@Top,@SQL_Where;

SELECT @Top = 1;
EXECUTE [dbo].[usp_Select_Data] @Team,@Table_Name,@Top,@SQL_Where;

SELECT @SQL_Where = '[cost] = 99.9999';
EXECUTE [dbo].[usp_Select_Data] @Team,@Table_Name,@Top,@SQL_Where;
```

usp_Update_Data

Updates individual values in a virtual table.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100), @Column_Name nvarchar(100), @RowID bigint,
@NewVal nvarchar(max);

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes', @Column_Name = 'Published', @RowID = 1, @NewVal =
'2013-05-16 17:55:00.555';

EXECUTE dbo.usp_Update_Data @Team, @Table_Name, @Column_Name, @RowID, @NewVal;
```

usp_Delete_Data

Deletes a row of data from virtual tables.

Usage

```
DECLARE @Team nvarchar(100), @Table_Name nvarchar(100), @RowID bigint;

SELECT @Team = 'Rigel', @Table_Name = 'MovieQuotes',@RowID = 1;

EXECUTE [dbo].[usp_Delete_Data] @Team, @Table_Name, @RowID;
```