# Introduction to SQL Server 2005/2008 and Transact SQL

## Week 4: Normalization, Creating Tables, and Constraints

Some basics of creating tables and databases

Steve Stedman - Instructor
Steve@SteveStedman.com

# Last Weeks Review

Introduction to SQL Server and Transact SQL - Week 3

- Querying Related Tables
- Join Syntax
- Inner Joins
- Left / Right Outer Join
- Cross Joins
- Unions
- Sub-Queries

# This Weeks Overview

Introduction to SQL Server and Transact SQL - Class 4

- Review of Last Week
- Overview Of Database Design
- 1st, 2nd and 3rd Normal Form
- Creating Tables
- Lab: Creating Tables
- Creating Relationships
- Lab: Creating Relationships
- Implementing Constraints

# 1. Database Design

- Design and Normalization
- Design Notes
- Design Steps
- Capacity Considerations

# Design and Normalization

- **Design**
  - The process of creating the data model of a database.
- **Normalization**
  - A systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics

# Design Notes

- Design for what you need
- Keep it simple – somebody is going to have to maintain it
- Keep in mind what you might need
- Consider future growth
- Know why you need this data
- Most importantly design for what you need

# Database Design Steps

- Determine what needs to be stored?
- Conceptual schema
  - How data relates to other data
- Logically Structure Data
  - May be done independent of the database architecture
- Physical Database Design
  - How is it all going to be arranged in this database?

# Capacity Considerations

- Determining the required size of the database
- Planning for growth
- Planning for load
- Variables may include
  - Customer acquisition rate
  - Historic data retention period
- How big can your database grow?

# Database Design

Lab Project

- Create a database design on paper to hold the following data
- Name (First and Last)
- Address, city, state, zip
- Phone 1, Phone 2
- Email Address

# End of Database Design

- Any Questions

- Next Topic - 1st, 2nd, and 3rd Normal Form

# 2. 1st, 2nd and 3rd Normal Form

Normalization

- Database Normalization
- Data Anomalies Caused by:
  - Update
  - Insertion
  - Deletion
- Brief History/Overview
- 1st Normal Form
- 2nd Normal Form
- 3rd Normal Form

# Database Normalization

- **The main goal of Database Normalization is to restructure the logical data model of a database to:**
  - Eliminate redundancy
  - Organize data efficiently
  - Reduce the potential for data anomalies.

# Data Anomalies

## Normalization helps reduce data anomalies

- Data anomalies are inconsistencies in the data stored in a database as a result of an operation such as update, insertion, and/or deletion
- Such inconsistencies may arise when have a particular record stored in multiple locations and not all of the copies are updated

- We can prevent such anomalies by implementing 7 different level of normalization called Normal Forms (NF)
- We'll only look at the first three.

# Data Anomaly Causes

- Data Anomalies Caused by:
  - Update
    - Examples - Phone number changes
  - Insertion
    - Another customer with the same name
  - Deletion
    - Are all of the duplicates removed

# Brief History / Overview

## Normalization

- Database Normalization was first proposed by Edgar F. Codd
- Codd defined the first three Normal Forms, which we'll look into 3, of the 7 known Normal Forms

- In order to do normalization we must know what the requirements are for each of the three Normal Forms that we'll go over.
- One of the key requirements to remember is that Normal Forms are progressive. That is, in order to have 3rd NF we must have 2nd NF and in order to have 2nd NF we must have 1st NF.

# 1st Normal Form

## The requirements

- **The requirements to satisfy the 1st NF:**
  - Each table has a primary key: minimal set of attributes which can uniquely identify a record
  - The values in each column of a table are atomic (No multi-value attributes allowed)
  - There are no repeating groups: two columns do not store similar information in the same table

# 1st Normal Form Example

Un-normalized Students table:

| Student# | AdvID | AdvName | AdvRoom | Class1 | Class2 |
|---|---|---|---|---|---|
| 123 | 123A | James | 555 | 102-8 | 104-9 |
| 124 | 123B | Smith | 467 | 209-0 | 102-8 |

Normalized Students table:

| Student# | AdvID | AdvName | AdvRoom | Class# |
|---|---|---|---|---|
| 123 | 123A | James | 555 | 102-8 |
| 123 | 123A | James | 555 | 104-9 |
| 124 | 123B | Smith | 467 | 209-0 |
| 124 | 123B | Smith | 467 | 102-8 |

# 2nd Normal Form

## The Requirements

- **The requirements to satisfy the 2nd NF:**
  - All requirements for 1st NF must be met
  - Redundant data across multiple rows of a table must be moved to a separate table
    - The resulting tables must be related to each other by use of foreign key

# 2nd Normal Form Example

Students table

| Student# | AdvID | AdvName | AdvRoom |
|---|---|---|---|
| 123 | 123A | James | 555 |
| 124 | 123B | Smith | 467 |

Registration table

| Student# | Class# |
|---|---|
| 123 | 102-8 |
| 123 | 104-9 |
| 124 | 209-0 |
| 124 | 102-8 |

# 3rd Normal Form

## The requirements

- The requirements to satisfy the 3rd NF:
  - All requirements for 2nd NF must be met.
  - Eliminate fields that do not depend on the primary key;
    - That is, any field that is dependent not only on the primary key but also on another field must be moved to another table

# 3rd Normal Form Example

Students table:

| Student# | AdvID | AdvName | AdvRoom |
|---|---|---|---|
| 123 | 123A | James | 555 |
| 124 | 123B | Smith | 467 |

Student table:

| Student# | AdvID |
|---|---|
| 123 | 123A |
| 124 | 123B |

Advisor table:

| AdvID | AdvName | AdvRoom |
|---|---|---|
| 123A | James | 555 |
| 123B | Smith | 467 |

# 3rd Normal Form Example Cont.

Students table:

| Student# | AdvID |
|---|---|
| 123 | 123A |
| 124 | 123B |

Registration table:

| Student# | Class# |
|---|---|
| 123 | 102-8 |
| 123 | 104-9 |
| 124 | 209-0 |
| 124 | 102-8 |

Advisor table:

| AdvID | AdvName | AdvRoom |
|---|---|---|
| 123A | James | 555 |
| 123B | Smith | 467 |

# Normal Forms
## Conclusion

- **We have seen how Database Normalization can decrease redundancy, increase efficiency and reduce anomalies by implementing three of seven different levels of normalization called Normal Forms.**
- **The first three NF's are usually sufficient for most small to medium size applications.**

# Normalization

Lab Project

- **Apply the 1st, 2nd and 3rd Normal Forms to the following table**

| Student# | SID | AdvName | AdvRoom | Class1 | Class2 |
|---|---|---|---|---|---|
| 123 | 123A | James | 555 | 102-8 | 104-9 |
| 124 | 123B | Smith | 467 | 209-0 | 102-8 |

# Normal Forms

Lab Project

- Using the handout and the database design from earlier
- Redesign the database making use of the 1st, 2nd and 3rd normal forms

# End of Topic Normal Forms

Normalization and 1st, 2nd and 3rd Normal Forms

- **Any Questions**

- **Next Topic - Creating Tables**

# 3. Creating Tables

Things to consider

- Topics to consider when creating tables
- Data Types
- Data Type Sizes
- Nullability
- Identity
- Computed Columns
- Creating a Table
- Sparse Columns (New in SQL Server 2008)

# Data Types

- Int
- Decimal / Numeric
- Float / Real
- Money
- Datetime
- Binary / VarBinary
- Others

# Character Data Types

- Char / Nchar
- Varchar(n), Varchar (max)
- Nvarchar(n), Nvarchar(max)
- Text, Ntext
  - Not allowed with many operations like join

# Specialized Data Types

- Bit (0, 1 or null)
- Timestamp
- Uniqueidentifier (16 bit GUID)
- Sql_Variant (can change depending on what is stored)
- XML

# Data Type Sizes

| Size | Storage | Values |
|---|---|---|
| **tinyint** | 1 byte | 0 to 255 |
| **smallint** | 2 bytes | -32,768 to 32,767<br>(64k or 64 thousand) |
| **int** | 4 bytes | -2,147,483,648 to 2,147,483,647<br>(about 4gb or 4 billion) |
| **bigint** | 8 bytes | −9,223,372,036,854,775,808 to<br>+9,223,372,036,854,775,807 |

# Nullability

- Null is not a value and does not consume storage in the same way as other values
- Different from 0 or the empty string “”

- use ISNULL ( check_expression , replacement_value ) to modify null values in output
- WHERE column_name IS NULL to test if a column is null

# Identity

- Does not allow nulls
- Automatically defaults to the next highest number in ascending sequence
- Cannot insert to an identity column

# Accessing the Identity Value

Sample

```
DECLARE @InsertedRows AS TABLE (Id int);
DECLARE @NewId AS INT;
INSERT INTO HumanResources.Employees
  ( /* column names */)
OUTPUT Inserted.Id INTO @InsertedRows
  VALUES (/* column values */);
SELECT @NewId = Id FROM @InsertedRows
```

# Computed Columns

- The value of the column is calculated on the fly based on other columns
- Calculated every time they are calculated unless PERSISTED
- Cannot be inserted or updated
- Example:
  - In the AdventureWorks sample database, the TotalDue column of the Sales. SalesOrderHeader table has the definition: TotalDue AS Subtotal + TaxAmt + Freight

# Creating a Table

- Need to know:
  - Where to create it (database, schema, storage)
  - Table Name
  - Columns, data type, and nullability
- Script it or use SSMS

# Creating Table Script (TSQL)

```
CREATE TABLE [HumanResources].[EmployeeAddress](
[EmployeeID] [int] NOT NULL,
[AddressID] [int] NOT NULL,
[ModifiedDate] [datetime] NOT NULL,
);
```

- Very Basic Create Table Syntax
  - There are many more options available

# Temporary Tables

See the Week 4 Random People Sample

- Similar to standard tables, but the scope and lifetime are different.
- Useful for functions, trigger or stored proc
- Scope
  - Global tables are available to everyone
  - Local tables can only be seen by the user who created them
- **Temporary tables can be global or local**
  - # pound sign denotes temporary table
  - ## double indicates global
  - @ or @@ denotes a table variable

# Lab Project - Creating Tables

Lab Project

- Using the database design from earlier in this class
- Using the SQL_Class database
- Create the tables that you designed earlier

# End of Topic Creating Tables

- Any Questions

- Next Topic - Creating Relationships

# 4. Creating Relationships

- What are Database Relationships
- Create a relationship in a database diagram
- Create a relationship in Table Designer
- Create the relationship with TRANSACT SQL

# What are Database Relationships

- Relationship between tables using:
  - Primary Keys
  - Foreign Keys

# Primary Keys

- A unique identifier that identifies a row in a table.
- Types of primary keys
  - Numeric
  - GUID’s (Globally Unique Identifiers)
  - Sequential GUID’s
  - Identity – unique throughout the database.
- Should be indexed for performance

# Foreign Keys

- A reference to another tables primary key
  - Must be the same data type
- Used to simplify a join between 2 or more tables
- Can provide abstraction
- Can be used for cascading deletes
- A table can contain multiple foreign keys providing relationships with multiple tables

# Create a relationship in Database Diagram

- In your database diagram, click the row selector for the database column or combination of columns that you want to relate to a column in another table.
- While the pointer is positioned over the row selector, click and drag to the related table.
- Release the mouse button. The Create Relationship dialog box appears and attempts to match the columns you selected with columns of the same name and data type in the related table.
- In the Create Relationship dialog box, confirm that the columns you want to relate are shown in the Primary key table and Foreign key table lists.

# Relationship with the Table Designer

- Open the Table Designer for the table that will be on the foreign key side of the relationship.
- Right-click in the Table Designer and choose Relationships.
- Click the New button.
- From the drop-down list in Primary Key Table, choose the table that will be on the primary-key side of the relationship. In the grid beneath, enter the columns contributing to the table's primary key. In the adjacent grid cell to the left of each column, enter the corresponding foreign-key column of the foreign-key table.
- The table designer suggests a name for the relationship. To change this name, edit the contents of the Relationship Name text box.
- Choose Close to create the relationship

# Creating Relationships

Lab Project

- For the tables created earlier, create the appropriate Primary and Foreign Keys

# End of Creating Relationships

- Any Questions

- Next Topic - Implementing Constraints

# 5. Implementing Constraints

- Constraints can be placed on tables or columns to implement business logic

- Types of constraints
  - Check Constraints
  - Rules
  - Default Constraints
  - Unique Constraints
  - Primary Keys
  - Foreign Key

# Check Constraints

- **Called on Insert and Update, but not called for Delete**
- **Used to validate single columns or multiple columns in a table**

- **Examples**
  - Validate Social Security Number format
  - Validate Email Address

# Rules

- Similar functionality as a check constraint, but stored separately
- Like a stored procedure or function
- Use sp_bindrule stored proc to bind it to a column

# Default Constraints

- Applies to one or more columns in a table
- Fills in a column if it is not specified on Insert
- Does not apply to Select, Update or Delete

# Unique Constraints

- Uses an index to enforce uniqueness
- No 2 rows in the table can contain the same value in the unique column
- Examples
  - Employee ID
  - Student ID
  - Social Security Number
  - Drivers License Number

# Primary Keys

- Columns or combination of columns to allow a rows to be uniquely identified
- Only one primary key per table
- Clustered Index automatically created for each primary key

- Good for integer
- Bad if uniqueidentifier

# Foreign Keys

- Links to a primary key in another table
- Value must exist in another table
- Prevents deleting of the referenced row or table

# End of Implementing Constraints

- Any Questions

- Next Topic - Lunch

# After Lunch

Technology Theme

## Class 5

- SQL Server Reporting Services
- SSRS Reporting
- Matrix Reporting
- Advanced Reporting
- Course Review

# Any Questions?

**Lunch Break**